Researching, Deploying and Applying the private cloud with Openstack platform

LUONG Vinh Thao

10/11/2014

Contents

Abstract

ACKNOWLEDGEMENTS

Chapter 1

INTRODUCTION

1.1 The schedule table

No.	Task	Start day	End day
1	Meeting and discussion about research project	2 June, 2014	7 June , 2014
2	Review services and deploment mode of cloud	9 June, 2014	14 June , 2014
3	Research openstack platform	16 June, 2014	30 June , 2014
4	Design the topology	1 June, 2014	21 July , 2014
5	Implement private cloud (compute, network)	22 July, 2014	15 Aug, 2014
6	Implement Storage	16 Aug, 2014	9 Sep, 2014
7	Automation the install by shell script	10 Sep, 2014	30 Sep, 2014
8	Build logging server	13 Oct, 2014	20 Oct, 2014
9	Build monitor server	20 Oct, 2014	31 Oct, 2014
10	Testing and demo	1 Nov,2014	13 Nov,2014
11	Report	15 Nov,2014	9 Dec,2014

1.2 The Report Structure:

The report comprises the 5 chapter: The Report Structure:

- Chapter 1. Introduction: Introduction about the Vietnam Nation University-VNU where i have carried out my internship
- Chapter 2. Background: the knowledge about cloud computing, the openstack platform and analysis request
- Chapter 3. Deployment: Deploy the topology and configure all the services. Writing the script to automate the install.

- Chapter 4. Demo: Demo create instance, network, volume, image, snapshot and test upload image into glance, upload file into swiff
- Chapter 5. Conclusion and Future work:

Chapter 2

BACKGROUND

2.1 Introduction Cloud Computing

Cloud computing is a computing model, refers to the on-demand delivery of IT resources and applications via the Internet with pay-as-you-go pricing. The cloud mode has 5 essential characteristics, 3 service models and 4 deployment models. In this chapter, I will talk about the property of cloud.

- Essential Characteristics:
  - On-demand self-service: A user can individually provision computing capabilities, such as server time and network storage and this process need automatically without requiring human interaction.
  - Broad network access : The availability access over the network and many different device allow access through standard mechanisms.
  - Resource pooling : All of resource are centralize to service for customer. And the tenant cant know the information about resource.
  - Rapid elasticity: Provisioning is rapid and scales out or is based on need. This characteristics increase the use of resource more flexible.
  - Measured service: The cloud need it to manage the service and billing. Beside, cloud automatic control and optimize resource use by leveraging a metering. Resource can be collected , monitored, reported and billing. This ensure the user can pay per use.
- Service Models:

  - Software as a Service(Saas): The capacity provide to the user use the software running on the cloud . The app can accessible from client through web interface , API or program interface.
  - Platform as a Service (PaaS): The capacity provided to the user deploy the application onto the cloud infrastructure base on programming languages, libraries, services or tools supported. The client does not manage or control the underlying cloud infrastructure like Sass but has control over the deployed applications and possibly configuration settings for the application-hosting environment infrastructure consumer-created. Some of PaaS is: Heroku, Google App Engine
  - Infrastructure as a Service (IaaS): The capacity provided the infrastructure such as instance or vitural machine to the consumer so that user can run the app or deploy operation system . Some of IaaS is : Amazon EC2, Google Compute Engine, Rackspace Cloud, Windows Azure, Digital Ocean.

- Deployment Models:
  - Private cloud: A private cloud operates for a single organization
  - Public cloud: It build to provide the service for anyone on the internet. Public cloud has an infrastructure that is available to the general public and is likely owned by a cloud services company.
  - Hybrid cloud: A hybrid cloud is a combine of public cloud and private cloud.

2.2 Introduction Openstack platform

2.2.1 What is Openstack

2.2.2 Openstack Workflow

The VM workflow:

(1) The user use the openstack dashboard - horizon to create the instance. (2) Horizon send HTTP request to keystone for authentication. (3) Keystone send token back to the horizon. (4) Horizon send POST request to Nova. It contain the token and request for create instance. (5) Nova send HTTP request to validate token to keystone. (6) If request has been validated, no action has been taken yet. (7) Nova call Neutron API to request provision network for instance. (8) Neutron configuration IP address, gateway, subnet,

DNS, etc. (9) Nova request the instance image from Glance via imageid. (10) If the imageid found, it send the URI back to nova and nova download image base on this URI.

2.2.3 Openstack Comute

2.2.4 Openstack Network

2.2.5 Openstack Storage

2.2.6 Openstack Dashboard

2.2.7 Openstack Shareservice

Chapter 3

DEPLOYMENT

3.1 Topology

3.2 Topology

3.3 Log Management

Logging is an important activity in all computer systems, and openstack is the same. It allows an administrator to track down problematic activity that can be used to help provide a solution.Understanding where the services log and managing those logs to allow administrator to identify problems quickly and easily. Understanding logging in openstack is important to ensure the environment is healthy and the able to submit log entries back to the community to help fix bugs. For log investigation, some tool like cat, tail, grep may be useful to diagnose and identity the problem in the log file. But what happen if we have a lot of servers and we want to log collection, processing and extracting data from the huge of log event. Some command above cannot solve this problem. All of the log event need to be centralize. Therefore, I will deploy the central log server to collect log event, parsing, indexing and storing to help the administrator track down the problem. There are many tool to do it but I need a solution easy to deploy, easy to operate, scale and open source. In this case, I choose some tool to do that. I install four component: Logstash, Elasticsearch, Kibana, Rsyslog (default install on ubuntu).

- Logstash: Receive log event from remote node and index the log. logstash is particularly useful since not all components follow the same format for log messages. logstash grok filter helps normalize various log messages into a format that can be indexed and searched.

- Elasticsearch: Store the log event and allow to search
- Kibana: Provide the web interface to easy monitor and handle the log.
- Rsyslog: send log event from service of openstack to logstash. It used to install on all of the remote node

3.3.1 Configure on the remote node

On the remote node, i must to send the log event to central node. First, i will enable openstack to send the rsyslog. Luckily, all of the service of openstack support rsyslog to send log event but the default is disable. To enable, we must change flag use_syslog = false to use_syslog = true on the configure file of services. There is example for keystone service.

```
sed -i "s/#verbose=false/verbose = False/" .../keystone.conf
sed -i "s/#debug=false/debug = False/" .../keystone.conf
sed -i "s/#use_syslog=false/use_syslog = True/" .../keystone.conf
sed -i "s/#syslog_log_facility=LOG_USER/syslog_log_facility
= LOG_LOCAL0/" /etc/keystone/keystone.conf
```

In this case the syslog facility is use to determined which log of service. For priority, i only want to get the **WARNNING** and **ERROR**. So, i must to use the two flag verbose = False and debug = False. This table is more detail about the priority on openstack.

Status	Flag	Log
verbose=true	debug=true	DEBUG, ERROR, WARNING, INFO
verbose=false	debug=true	DEBUG, ERROR, WARNING, INFO
verbose=true	debug=false	WARNING, INFO
verbose=true	debug=false	WARNING, INFO

3.3.2 Configure on the central log node

On the central node, I install Logstash, elasticsearch and kibana and configure that to catch, index and store log event. First, I configure Rsyslog to open the tcp port 9000 to receive log event from all nodes.

```
local0.*  @@logserver:9000  #nova
local1.*  @@logserver:9000  #glance
local2.*  @@logserver:9000  #keystone
```

```
 local3.*  @@logserver:9000  #cinder
 local4.*  @@logserver:9000  #neutron
 local5.*  @@logserver:9000  #swiff
 local6.*  @@logserver:9000  #ceilometer
```

Second, I configure logstash to receive log event, parsing and storing. I open the port 9000 to receive log event.

```
input {
tcp {
     port => 9000
     type => syslog
   }
}
```

Third, I use the grok filter to parsing the log line.

```
filter {
        if [type] == "syslog" {
        grok {
                patterns_dir=> "/opt/logstash/patterns"
                match => ["message","%{OPENSTACK_SYSLOG}"]
                add_tag => ["OpenStack"]
            }
        }
```

The struct of log on service of openstack look like :

```
<134> 2014-09-30T16:09:18.580221+07:00 controller
2014-09-30 16:09:18.579 1261 INFO oslo.messaging._drivers.impl_rabbit
 [-] Connected to AMQP server on controller:5672.
```

So, I want extra data like that.

```
Timestamp  : 2014-09-30T16:09:18.580221+07:00
Hostname    : controller
Timestamp   : 2014-09-30 16:09:18.579
PID      : 1261
LOGLEVEL    : WARNING
PROGRAM    : oslo.messaging._drivers.impl_rabbit
ID        : [-]
MESSAGE     : Connected to AMQP server on controller:5672
```

Here, logstash support grow pattern to help parsing the log event. Therefore, i write a grow pattern to do that.

```
OPENSTACK_LOGLEVEL ([A-a]lert|ALERT|[T|t]race|TRACE|
[D|d]ebug|DEBUG|[N|n]otice|NOTICE|[I|i]nfo|INFO|[W|w]arn?(?:ing)?
|WARN?(?:ING)?|[E|e]rr?(?:or)?|ERR?(?:OR)?|[C|c]rit?(?:ical)?
|CRIT?(?:ICAL)?|[A|a]udit|AUDIT|[F|f]atal|FATAL|
[S|s]evere|SEVERE|EMERG(?:ENCY)?|[Ee]merg(?:ency)?)
OPENSTACK_LOG %{TIMESTAMP_ISO8601:OS_timestamp} %{POSINT:OS_pid}
%{OPENSTACK_LOGLEVEL:OS_loglevel} %{PROG:OS_program}
(?<OS_id>\[(?:(req-%{UUID}|%{UUID} |%{BASE16NUM}|None|-|%{SPACE}))+\])?
%{GREEDYDATA:OS_message}
OPENSTACK_LOGSTORAGE %{GREEDYDATA:OS_message}
OPENSTACK_LOGLINE %{OPENSTACK_LOG}| %{OPENSTACK_LOGSTORAGE}
OPENSTACK_SYSLOG(?<sys_pri>\<%{POSINT}\>)%{TIMESTAMP_ISO8601:timestamp}
%{IPORHOST:OS_hostname} %{OPENSTACK_LOGLINE}
```

Finnaly, outputs the result of the second step to ElaticSearch

```
output {
elasticsearch { host => localhost }
stdout { codec => rubydebug }
}
```

3.4 Monitor

“ 80% of the mean time to repair is wasted on trying to locate the issue ” - gartner. That a reason, I need to monitor and handle the error of the system. In Openstack, I must monitor and maintain a lot of server and services. In particular, some of services must be high availability such as neutron, nova metadata, DHCP agent ..... The Openstack didn't support services to do that. Consequently, I will zabbix to monitoring the healthy of system. All of them are open source software. . This solution is very useful to identity problem from server or services. The aim of zabbix is monitoring the status of all nodes such as ram, CPU, memory, disk,..etc. and alert the notification when something to error.

Openstack platform has the ceilometer service to monitor the status of instance. But this service didn't monitor the physical server. Openstack is Monitor as a service (MOaaS) are still at “ blueprint ” state. Consequently, I need to monitor the healthy of server and alert me when the problem happen . There are many open source software can do it such as: nagios, Zabbix, Zennos ...etc. But the monitor solution must be montioring the physical hardware, the application run on it and the virtual machine. In this case, i choose Zabbix because it is a enterprise-class open source distributed monitoring solution and easy to deploy, easy to monitor, the lower hardware than another tool. Beside, Zabbix has the real-time graphing to monitor items by immediately graphed. Zabbix has four compoment: Zabbix Server, Zabbix proxy, Zabbix Agent, Web Interface. Zabbix proxy is a cache for zabbix server and very useful in the big network. But in this case, i will not deploment that.

Zabbix has four component: Zabbix Server, Zabbix proxy, Zabbix Agent, Web Interface. Zabbix proxy is a cache for zabbix server and very useful in the big network. But in this case, i will not deployment that.

- Zabbix Server: It's a central component and responsible for receive data from agent and store all configuration, statistical and operational data

- Zabbix Agent: It's deployed on the remote node and collect the data to send Zabbix server.

- Web Interface: It's provide to access zabbix from anywhere and any platform. Beside the web interface is very useful to manage and configure without using any command.

Chapter 4

DEMO

4.1 Create Instance

4.2 Create Volume

4.3 Upload file

Chapter 5

CONCLUSION AND FUTURE WORK

Bibliography

[1] Rektorys, K., *Variational methods in Mathematics, Science and Engineering*, D. Reidel Publishing Company, Dordrecht-Hollanf/Boston-U.S.A., 2th edition, 1975

[2] Bertóti, E.: *On mixed variational formulation of linear elasticity using nonsymmetric stresses and displacements*, International Journal for Numerical Methods in Engineering., **42**, (1997), 561-578.

Appendix A

Script for automate install