

SaaS OverCloud 기술 문서 #6

OverCloud를 위한 UnderCloud 종속성 분석을 통한 간편한 마이그레이션 기능 연구

Document No. SaaS OverCloud #6
Version 0.1
Date 2016-10-14
Author(s) GIST Team

■ 문서의 연혁

버전	날짜	작성자	비고
초안 - 0.1	2016. 10. 14	Jargalsaikhan Narantuya, 김성환, 장한이	
0.2	2016. 10.		
0.3	2016. 10.		
0.5			
0.7			
0.8			
0.9			
1.0			

본 문서는 2016년도 정부(미래창조과학부)의 재원으로
정보통신기술진흥센터의 지원을 받아 수행된 연구임 (No.R7117-16-0218,
이중 다수 클라우드 간의 자동화된 SaaS 호환성 지원 기술 개발)

The reserach was supported by Institute for Information &
communications Technology Promotion(IITP) grant funded by the
Korea government(MSIP) (No.R7117-16-0218, Development of
Automated SaaS Compatibility Techniques over Hybrid/Multisite
Clouds)

Contents

SaaS OverCloud #6. OverCloud를 위한 UnderCloud 종속성 분석을 통한 간편한 마이그레이션 기능 연구

1. 종속성 분석을 통한 간편한 마이그레이션 기능 연구	5
1.1. 목적 및 개요	5
1.2. Dependency	6
1.2.1. Dependency detection strategies	7
1.3. 마이그레이션을 위한 멀티사이트 Private OpenStack 설치/설정	10
1.3.1. 시스템 요구사항	10
1.3.2. GitHub를 이용하여 설치 스크립트 내려 받기	13
1.3.3. OpenStack 설치하기	14
1.3.4. OpenStack 검증하기	15

표 목차

그림 목차

그림 1 멀티사이트 환경에서의 마이그레이션 기술 연구	5
그림 2 Traffic amongst 16 VMs	7
그림 3 Similar Spikes in CPU Usage of Dependent VMs	8
그림 4 OpenStack 설치를 위한 최소 요구사항	10
그림 5 마이그레이션 테스트베드 구성도	10
그림 6 가상화 기술 지원 여부 확인	11
그림 7 가상화 기술 (Virtualization Technology) Enable	11
그림 8 VMware에서 Host-only 방식의 Network adapter 추가하기	12
그림 9 OpenStack 설치 스크립트 목록	13

SaaS OverCloud #6.

OverCloud를 위한 UnderCloud 종속성
분석을 통한 간편한 마이그레이션 기능 연구

1. 종속성 분석을 통한 간편한 마이그레이션 기능 연구

1.1. 목적 및 개요

- 본 문서에서는 아래 그림 1과 같이 멀티사이트 환경에서 종속성 관계에 따라 마이그레이션을 수행하기 위한 개념 및 인프라 환경, 즉 멀티사이트 Private OpenStack [1]에 대한 구현에 대한 상세 내용을 기술한다.

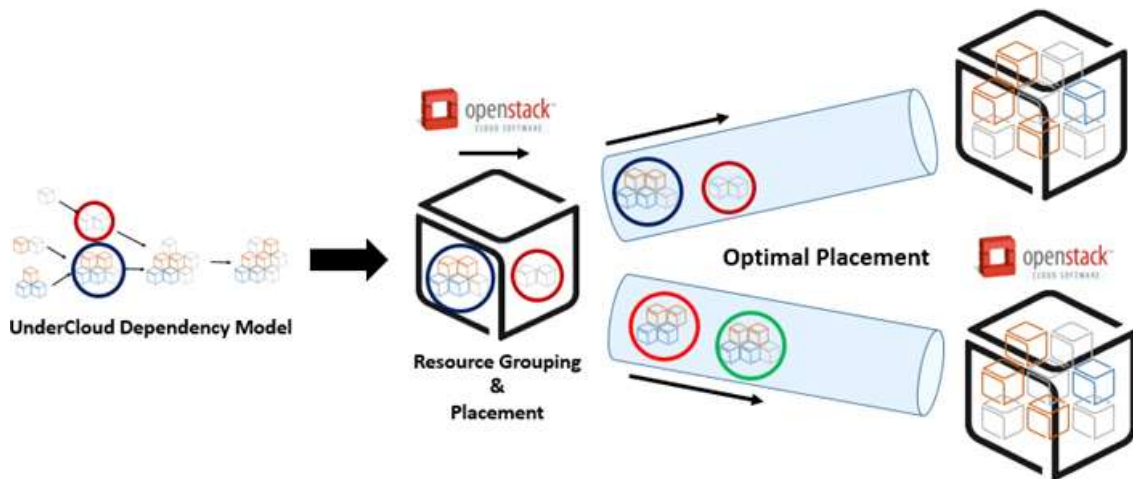


그림 1 멀티사이트 환경에서의 마이그레이션 기술 연구

- 여러 개의 박스가 긴밀히 상호작용하는 클라우드를 마이그레이션하기 위해서는 각각의 박스의 대한 종속성 관계에 대한 분석이 선행되어야 한다. 이를 위해 먼저 SmartX 공용개발환경을 사용하여 멀티사이트 Private OpenStack을 구축 후 박스 간 트래픽 정보를 통해 클라우드 내 박스 간 종속성(Dependency)을 기술하기 위한 수학적 모델을 도출한다.

1.2. Dependency

- o In cloud datacenters, applications are hosted among one or more VMs, thus they can be classified into two classes, namely single and multi VM applications. The running of multi-VM applications will bring with communications between VMs in real-time, generating communication dependency and showing up traffic flows passing through the shared network, which naturally is a communication dependency between VMs. A VM hosting a single-VM application has not communication dependency with other VMs. Moreover, some cloud customers make their VMs having communication with other applications. For instance, some applications use a data that is generated by another application. In this regard, these VMs with affinity are derived from user-defined communication dependency.
- o For a datacenter administrator, to efficiently manage a datacenter such as allocating resources, migrating VMs between physical machines, and detecting and mitigating critical faults, it is important to be able to identify the dependencies that exist between the VMs. This is because, by knowing which VMs are dependent, an administrator can make more sophisticated decisions including, but not limited to the following list:
 - Migrating VMs to another physical machine: if two VMs are dependent on one another, it might be preferable to move them together.
 - Whether relinquishing resources from some VM may affect another dependent service: allocation and management of resources can be made more intelligent. It might seem like a VM is overprovisioned, but the service of this VM may be used by multiple dependent VMs.
 - Identifying causes of faults: based on the knowledge of dependencies, the management system can better determine cause-effect relations, and determine how a fault may manifest itself.

1.2.1. Dependency detection strategies

- o **Traffic based detection:** To find a dependency between VMs, some works analyze the traffic fingerprinting results to extract dependency of VMs. They capture network traffic over the NIC of each physical machine which is hosting virtual machines. Each traffic record signifies a network packet transferred from a source server to a destination server. It is a triple like $\langle \text{source ID, destination ID, volume} \rangle$, in which the source ID and destination ID denote the source VM IP and destination VM IP address, respectively, and the volume denotes the total summed packet size (bytes) of traffic captured within a period of monitoring time.

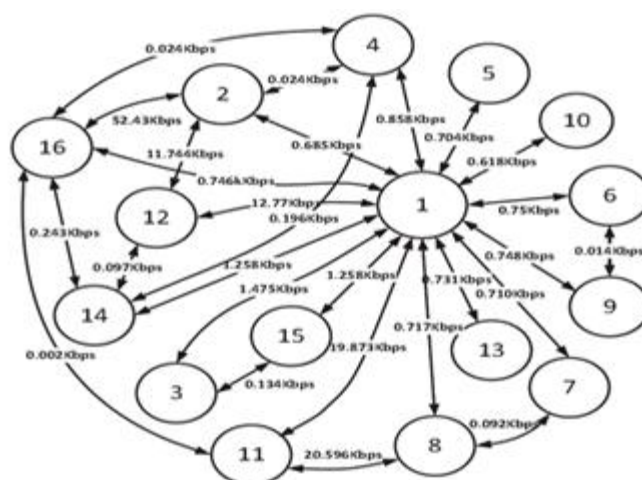


그림 2 Traffic amongst 16 VMs

- Step 1.** Input a VM set V with the number N , and a VM-affinity relation set VR with the number E .
- Step 2.** Initially each VM of V is built as a VM-affinity group set AG (singleton set).
- Step 3.** For each VM in a VM-affinity relation of AR , find the VM-affinity group set which contains the VM, and get two VM-affinity group sets.
- Step 4.** Apply union operation to the two VM-affinity group sets to get a new VM-affinity group set.
- Step 5.** If there are any other VM-affinity relations, then go to Step 3; otherwise go to Step 6.
- Step 6.** Output all the VM-affinity group sets as final grouping result.

For instance: $V = \{a, b, c, d, e, f, g\}$, $AR = \{\{a \sim b\}, \{b \sim d\}, \{c \sim f\}, \{f \sim g\}\}$, $a \sim b$: a and b have a relation

Initial VM-affinity group set $AG = \{\{a\}, \{b\}, \{c\}, \{d\}, \{e\}, \{f\}, \{g\}\}$

1. $AR = \{\{a \sim b\}, \{b \sim d\}, \{c \sim f\}, \{f \sim g\}\}$: $AG = \{\{a, b\}, \{c\}, \{d\}, \{e\}, \{f\}, \{g\}\} \Rightarrow AG = \{\{a, b\}, \{c\}, \{d\}, \{e\}, \{f\}, \{g\}\}$

2. $AR = \{\{b \sim d\}, \{c \sim f\}, \{f \sim g\}\}$: $AG = \{\{a, b, d\}, \{c\}, \{e\}, \{f\}, \{g\}\} \Rightarrow AG = \{\{a, b, d\}, \{c\}, \{e\}, \{f\}, \{g\}\}$

3. $AR = \{\{c \sim f\}, \{f \sim g\}\}$: $AG = \{\{a, b, d\}, \{c, f\}, \{e\}, \{g\}\} \Rightarrow AG = \{\{a, b, d\}, \{c, f\}, \{e\}, \{g\}\}$

4. $AR = \{\{f \sim g\}\}$: $AG = \{\{a, b, d\}, \{c, f, g\}, \{e\}\} \Rightarrow AG = \{\{a, b, d\}, \{c, f, g\}, \{e\}\}$

5. $AR = \{\}$ $\Rightarrow AG = \{\{a, b, d\}, \{c, f, g\}, \{e\}\}$

Final affinity group is $AG = \{\{a, b, d\}, \{c, f, g\}, \{e\}\}$

- o **Resource utilization based detection:** Multi VM applications, dependent VMs, have request-response type of interactions, where a client VM makes a request to a server which performs some computation and responds. In a request-response type of architecture, when a client's workload increases, it makes more requests to the server it depends on, and the server's workload increases. In [2], they expect to see a similar spike in the CPU utilization of both the client and server as illustrated in following figure.

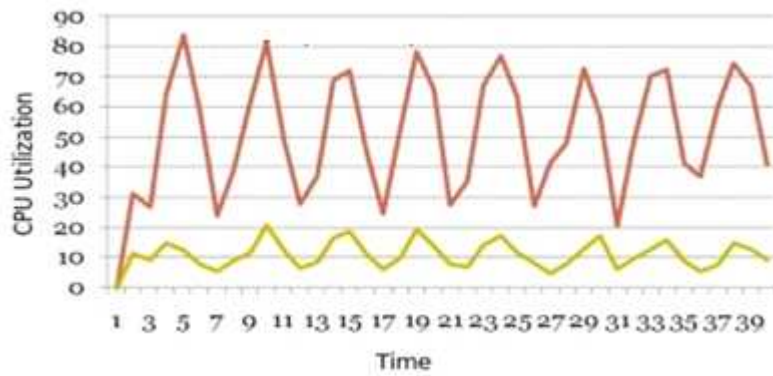


그림 3 Similar Spikes in CPU Usage of Dependent VMs

They model each VM's CPU utilization as a time series signal, and use this information to classify VM communication dependency.

References:

1. Chen, J., Chiew, K., Ye, D., Zhu, L., & Chen, W. (2013, March). **Aaga: Affinity-aware grouping for allocation of virtual machines**. In Advanced Information Networking and Applications (AINA), 2013 IEEE 27th International Conference on (pp. 235-242). IEEE.
2. Apte, R., Hu, L., Schwan, K., & Ghosh, A. (2010, June). **Look Who's Talking: Discovering Dependencies between Virtual Machines Using CPU Utilization**. InHotCloud.

1.3. 마이그레이션을 위한 멀티사이트 Private OpenStack 설치/설정

1.3.1. 시스템 요구사항

- o OpenStack cloud를 구축하기 위해 OpenStack 매뉴얼에서 제시하는 최소의 시스템 요구사항은 그림 4와 같다.

- Controller Node: 1 processor, 2 GB memory, and 5 GB storage
- Network Node: 1 processor, 512 MB memory, and 5 GB storage
- Compute Node: 1 processor, 2 GB memory, and 10 GB storage

그림 4 OpenStack 설치를 위한 최소 요구사항

- o 한 사이트 내의 OpenStack node 간의 연결을 위하여 제어용(Control)과 데이터용(Data)의 목적에 맞춰 네트워크 분리(즉 망분리)를 실제로 제공하고, Node 들을 이에 맞춰서 분리된 제어, 데이터용 네트워크들과 연결해야 한다. VM 이미지가 이동하는 마이그레이션 경로에는 10G 네트워크를 사용하였다.

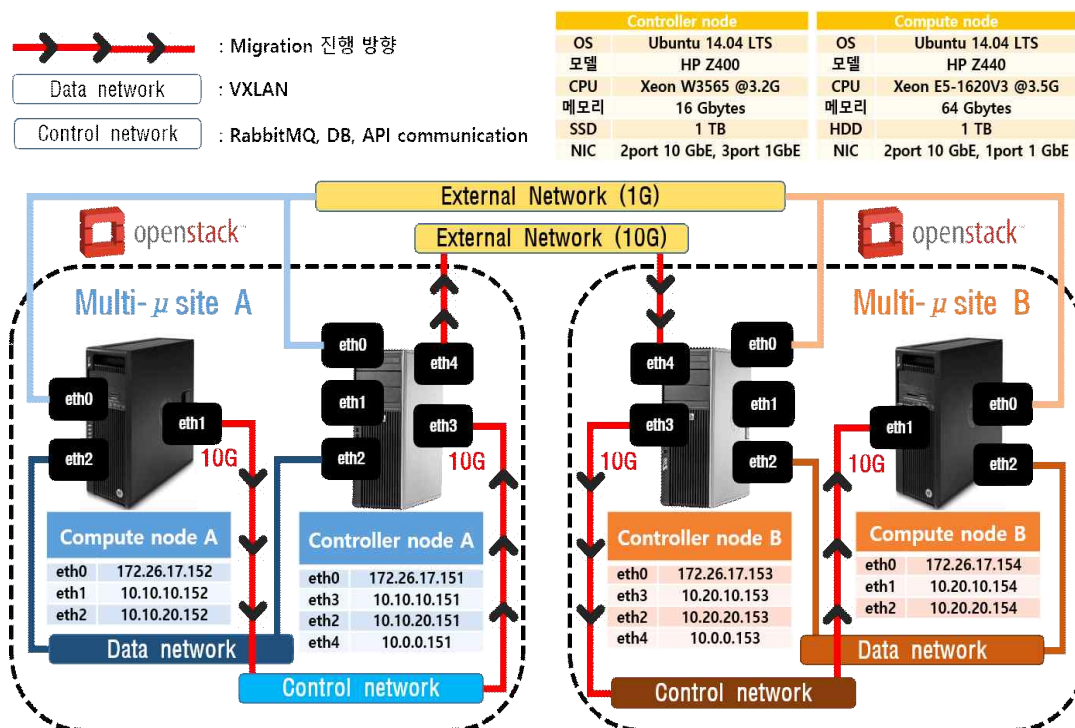
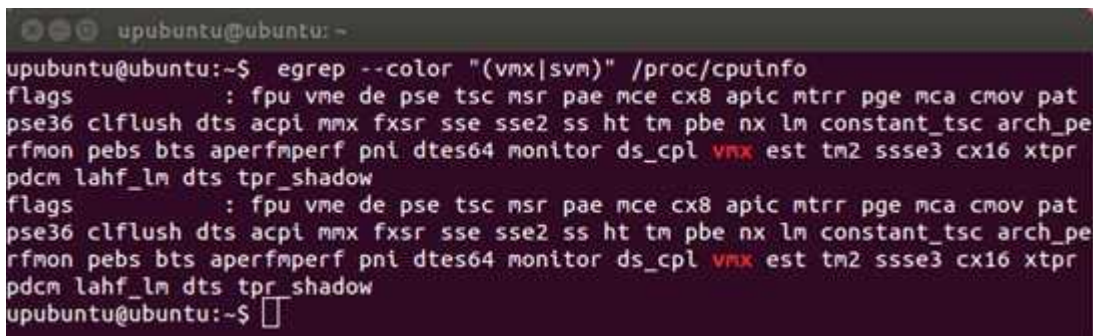


그림 5 마이그레이션 테스트베드 구성도

- o Compute node가 Hypervisor를 통해 VM을 실행시키려면 해당 PC의 CPU가 가상화 기술을 지원해야 한다. Ubuntu Linux의 경우 이를 확인하는 방법은 다음과 같다. 결과 화면에서 Intel CPU의 경우 vmx, AMD CPU의 경우 svm을 확인하면 된다.

```
$ sudo egrep '(vmx|svm)' /proc/cpuinfo
```

위 명령어의 실행 결과는 다음과 같다.



```
upubuntu@ubuntu:~$ egrep --color "(vmx|svm)" /proc/cpuinfo
flags              : fpu vme de pse tsc msr pae mce cx8 apic mtrr pge mca cmov pat
pse36 clflush dts acpi mmx fxsr sse sse2 ss ht tm pbe nx lm constant_tsc arch_pe
rfmon pebs bts aperfmperf pni dtes64 monitor ds_cpl vmx est tm2 ssse3 cx16 xtpr
pdcmlahf_lm dts tpr_shadow
flags              : fpu vme de pse tsc msr pae mce cx8 apic mtrr pge mca cmov pat
pse36 clflush dts acpi mmx fxsr sse sse2 ss ht tm pbe nx lm constant_tsc arch_pe
rfmon pebs bts aperfmperf pni dtes64 monitor ds_cpl vmx est tm2 ssse3 cx16 xtpr
pdcmlahf_lm dts tpr_shadow
upubuntu@ubuntu:~$
```

그림 6 가상화 기술 지원 여부 확인

- o 만일 vmx와 svm 모두 출력되지 않는다면 BIOS 메뉴에서 가상화 기술을 지원하고 있는지 확인한다. HP PC의 경우 BIOS 메뉴 상단의 Security -> System Security -> Virtualization Technology의 속성값을 Enable로 바꿔주면 된다.

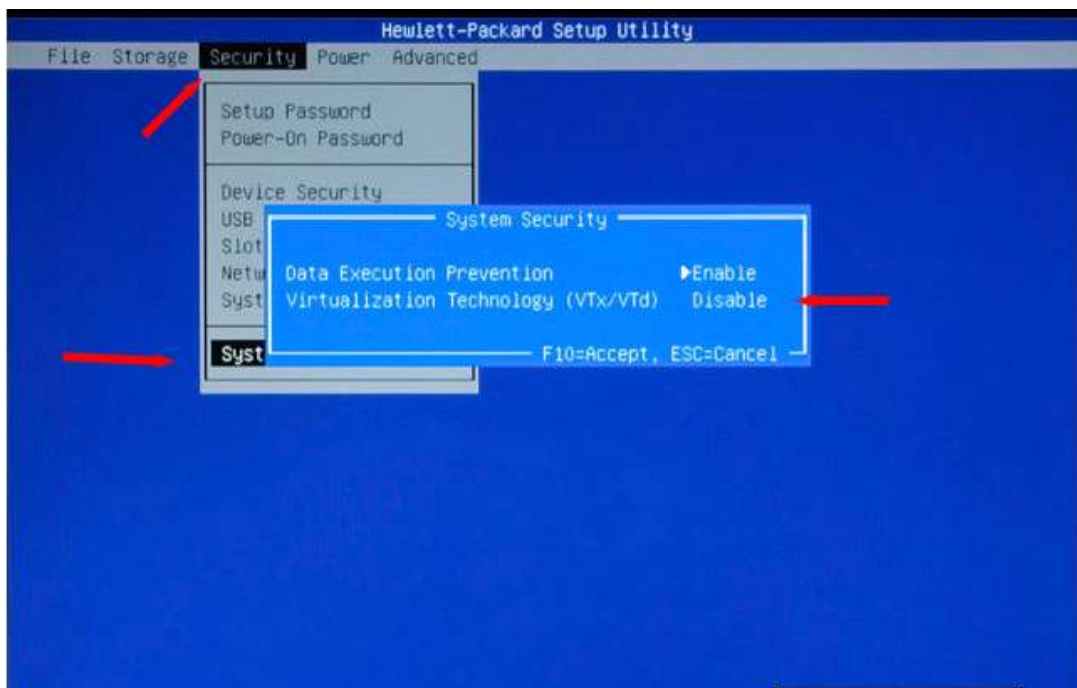


그림 7 가상화 기술 (Virtualization Technology) Enable

- o 실제 하드웨어 장비를 이용한 실험에서는 Controller와 Compute node 간의 네트워크를 Router나 Switch와 같은 장비를 이용하여 추가할 수 있으나 VM을 이용하여 구축하는 경우 VM에 대해 Network Adapter를 추가해주어야 한다. 아래 그림 8의 경우 VMware[2]의 설정 메뉴를 보여주고 있다. Host-only 방식의 Network Adapter를 추가로 구성하면 VM 간의 통신은 가능하나 외부와 단절되는 네트워크를 구성할 수 있다.

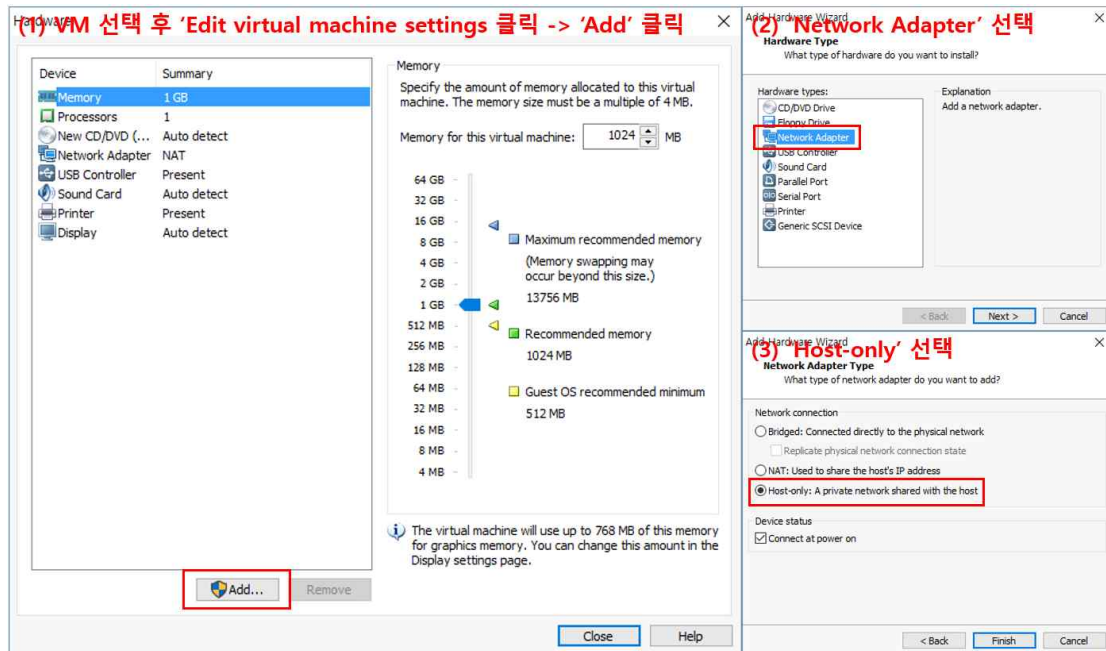


그림 8 VMware에서 Host-only 방식의 Network adapter 추가하기

1.3.2. GitHub를 이용하여 설치 스크립트 내려 받기

- o GitHub[3]는 대표적인 오픈 소스 코드 저장소이다. 본 실험에서는 OpenStack 설치 과정을 스크립트로 작성하여 자동화시켰다. 이 스크립트들은 현재 GitHub에 저장되어있는 상태이다. 각 스크립트를 설치 환경에 맞게 수정하여 명령어창에서 `sudo` 권한으로 실행시켜 주면 설치할 수 있다.

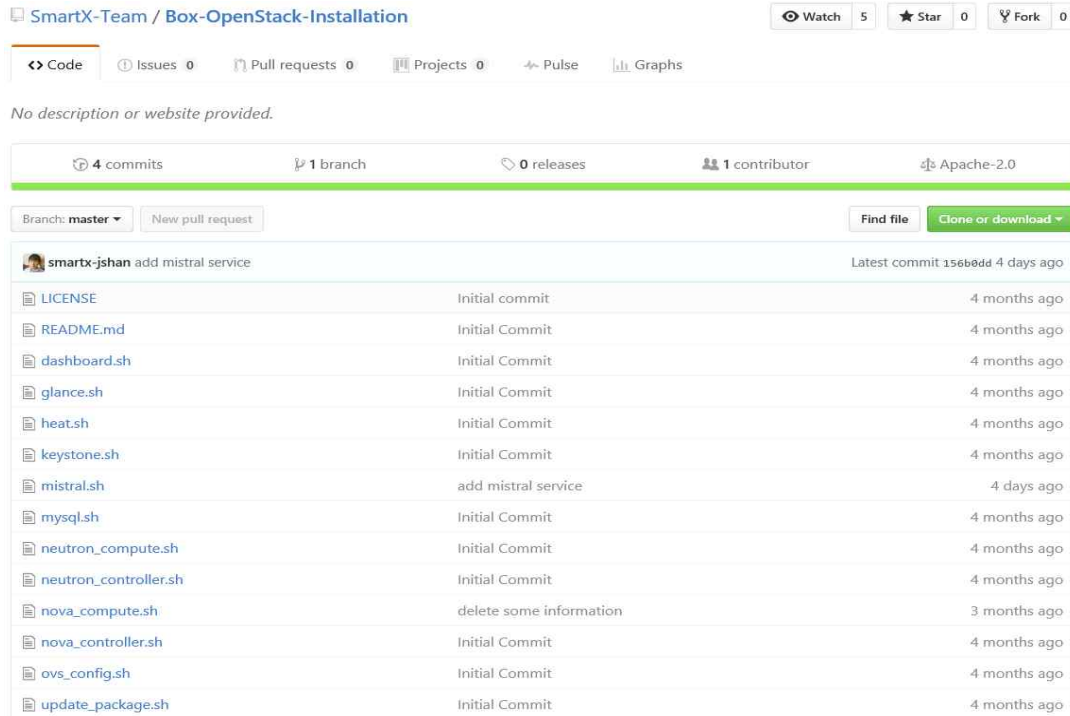


그림 9 OpenStack 설치 스크립트 목록

- o GitHub에 저장되어있는 스크립트를 통해 로컬 PC에서 작업하려면 먼저 GitHub에 가입한 후, 'git'을 설치한다.

```
$ sudo apt-get install git
```

이제 'git'에 자신을 소개할 차례이다. "You Name Here"에 자신의 이름을, "your_email@youremail.com"에는 GitHub에 가입할 때 사용한 이메일 주소를 입력한다.

```
$git config --global user.name "Your Name Here"
```

```
$git config --global user.email "your_email@yourmail.com"
```

다음으로 스크립트를 갖고 있는 저장소를 복사한다.

```
$git clone https://github.com/smartx-team/box-openstack-installation
```

1.3.3. OpenStack 설치하기

- o 'git'을 통해 받은 모든 설치 스크립트에는 아래와 같이 IP 주소가 설정되어 있다. 만약 2개의 네트워크 인터페이스를 이용하는 경우 M_IP와 C_IP를 동일하게 설정하도록 한다. 'M'은 Management, 'C'는 Control, 'D'는 Data를 뜻한다.

```
M_IP = 10.10.1.107  
C_IP = 10.10.10.107  
D_IP = 10.10.20.107
```

'nova_compute.sh'와 'neutron_compute.sh'의 경우 Compute node에서만 설치되는 스크립트이며 아래와 같이 총 5개의 IP 주소가 설정되어 있다. CTR은 'Controller'를 뜻한다. Controller에 해당하는 주소를 입력하도록 한다.

```
M_IP = 10.10.1.107  
C_IP = 10.10.10.107  
D_IP = 10.10.20.107  
CTR_M_IP = 10.10.10.10  
CTR_C_IP = 10.10.10.10
```

'ovs_config.sh'의 경우 아래와 같이 'INTERFACE'라는 변수에 'br-ex'라는 값으로 입력되어 있다. 이를 실제 포트 이름으로 수정하도록 한다. (ex. eth0, eth1, ...)

```
INTERFACE = br-ex
```

스크립트 실행 순서는 다음과 같다.

```
Controller node : update_package.sh -> mysql.sh -> keystone.sh -> glance.sh ->  
nova_controller.sh -> neutron_controller.sh -> ovs_config.sh -> heat.sh -> dashboard.sh
```

```
Compute node : update_package.sh -> nova_controller.sh -> neutron_controller.sh -> ovs_config.sh
```


1.3.4. OpenStack 검증하기

- o OpenStack의 서비스 검증, Node간의 연결 검증

References

- [1] OpenStack, <http://openstack.org>
- [2] VMware, <http://www.vmware.com>
- [3] GitHub, <http://github.com>

SaaS OverCloud 기술 문서

- 광주과학기술원의 확인과 허가 없이 이 문서를 무단 수정하여 배포하는 것을 금지합니다.
- 이 문서의 기술적인 내용은 프로젝트의 진행과 함께 별도의 예고 없이 변경될 수 있습니다.
- 본 문서와 관련된 대한 문의 사항은 아래의 정보를 참조하시길 바랍니다.
(Homepage: <https://smartx.kr>, E-mail: contact@smartx.kr)

작성기관: 광주과학기술원
작성년월: 2016/10