오픈 SmartX 플랫폼에 통합된 SmartX Box의 원격 자동설치 및 제어

OF@KOREN Document No. 1
Version 1.0
Date 2015-12-11
Author(s) GIST OF@KOREN 팀

■ 문서의 연혁

버전	날짜	작성자	비고
초안	2015. 10. 16	GIST OF@KOREN 팀	
1차 수정	2015. 11. 11	GIST OF@KOREN 팀	
2차 수정	2015. 11. 26	GIST OF@KOREN 팀	
3차 수정	2015. 12. 08	GIST OF@KOREN 팀	
1.0	2015. 12. 11	GIST OF@KOREN 팀	

본 연구는 한국정보화진흥원(NIA)의 미래네트워크연구시험망(KOREN) 사업 지원과제의 연구결과로 수행되었음 (15-951-00-001)

This research was one of KOREN projects supported by National Information Society Agency (15-951-00-001)

Contents

1. SmartX Box 및 가상놀이터 개요 ····· 1
1.1. 목적 및 개요 ····· 1
1.2. SmartX Box ····· 2
1.3. SmartX Virtual Playground ····· 4
1.4. 자동 Provisioning의 필요성 ····· 5
2. SmartX Provisioning Framework ····· 7
2.1. SmartX Provisioning Framework 개요 ····· 7
2.2. SmartX Provisioning Framework 요구사항 ····· 8
2.3. SmartX Provisioning Framework 소프트웨어 구현 ····· 9
2.4. SmartX Provisioning Framework 동작과정 설명 ····· 10
2.5. SmartX Provisioning Framework SW 설정 및 실행방법 ····· 13
3. SmartX Playground를 위한 Framework 활용사례 및 설정방법 ····· 17
3.1. SmartX Provisioning Framework의 활용을 위한 Use Case ····· 17
3.2. Ubuntu MAAS 개요 및 설정 ····· 19
3.2.1. Ubuntu MAAS 개요 ····· 19
3.2.2. Ubuntu MAAS 설정 ····· 21
3.2.3. 융합형 자원 박스 전원관리 설정 ····· 24
3.2.4. 원격 자동설치 도구 준비 및 설정 ····· 26
3.3. DevStack 버전 오픈스택 Installer 개요 및 상세 ····· 27
3.3.1. DevStack 개요 ····· 27
3.3.2. DevStack 버전 오픈스택 Installer 상세 ····· 29
3.4. Full 버전 오픈스택 Installer 개요 및 상세 ····· 30
3.4.1. 개요 ····· 30
3.4.2. Full 버전 오픈스택 Installer 상세 ····· 31
4. SmartX Provisioning Framework SW 검증 ····· 33
4.1. Ubuntu + OpenStack 설치 시간 측정 ····· 33

그림 목차

그림 1 융합형 SmartX Box 및 Box/Inter-Connect/Function의 3대 요소 ····· 3
그림 2 OF@KOREN Playground 구성도 ····· 4
그림 3 SmartX Provisioning Framework 소프트웨어 구조 ····· 9
그림 4 SmartX Provisioning Framework Working Procedure ····· 11
그림 5 Framework 내 박스 설정 파일 규격 ····· 13
그림 6 Playground Template 기본 포맷 ····· 14
그림 7 Playground Template 작성 예시: 5개 개별 클라우드 환경 ····· 14
그림 8 Playground Template 작성 예시: 2개 멀티노드 클라우드, 1개 Ubuntu ····· 15
그림 9 새로운 Installer 추가시 prepare_services.sh에 추가할 코드 양식 ····· 16
그림 10 오픈스택 클라우드의 논리적 아키텍쳐 ····· 17
그림 11 Ubuntu MAAS Web UI ····· 19
그림 12 Ubuntu MAAS Architecture ····· 20
그림 13 Ubuntu MAAS Working Procedure ····· 20
그림 14 Ubuntu CD를 통한 Ubuntu MAAS 설치 ····· 21
그림 15 Ubuntu MAAS Cluster 탭 화면 ····· 22
그림 16 Ubuntu MAAS 인터페이스 DHCP 설정 ····· 23
그림 17 Ubuntu MAAS SSH 키 등록 ····· 23
그림 18 Intel RMM 계정 등록 ····· 24
그림 19 Ubuntu MAAS 노드 Power 설정 ····· 26
그림 20 박스의 고정 IP 위한 DHCP 설정 ····· 27
그림 21 Devstack local.conf file (좌: Controller Node, 우: Compute Node) ····· 29
그림 22 DevStack 버전 오픈스택 Installer를 위한 Playground Template 양식 ····· 30
그림 23 Full 버전 오픈스택 Installer를 위한 Playground Template 양식 ····· 31
그림 24 Full 버전 오픈스택 Installer 설치/설정과정 ····· 32
그림 25 원격 자동 설치 검증 시나리오 ····· 33
그림 26 SmartX Automation Center UI ····· 33
그림 27 Ubuntu 설치 화면 ····· 34
그림 28 기본 설정 및 OpenStack 설치 시 화면 ····· 34
그림 29 설치 완료 시 화면 ····· 35
그림 30 설치 결과 검증 ····· 35
그림 31 자동 설치된 OpenStack의 Horizon Dashboard ····· 36
그림 32 자동 설치된 OpenStack 내 VM 생성 검증 ····· 36

1. SmartX Box 및 가상놀이터 개요

1.1. 목적 및 개요

o 본 문서에서는 KOREN 네트워크 상 OF@KOREN Playground (i.e. Testbed)에 배포되어 운용 중인 SmartX Box를 대상으로 자동화된 Provisioning 기능을 제공하는 SmartX Provisioning Framework의 설계, 구현 및 실제 활용 방법에 대해 상세히 기술한다. SmartX Provisioning Framework는 중앙에서 원격지에 분산되어 위치한 초 융합형 자원 박스들을 대상으로 DevOps[1] 관점에서 소프트웨어로 자동화된 설치/설정/제어하는 기능을 OF@KOREN Playground Operator 에게 제공한다. 본 Framework를 통해 OF@KOREN Playground를 소프트웨어 정의 인프라(Software-Defined Infrastructure) 관점으로 운영이 가능하다.

o 박스 간의 연결을 위하여 관리용(Management), 제어용(Control)과 데이터용(Data)의 목적에 맞춰 네트워크 분리(즉 망분리)를 실제로 제공하고, 융합형 Box들을 이에 맞춰서 분리된 관리, 제어, 데이터용 네트워크들과 연결해야 한다. 또한 박스가 분산되어 있는 환경에서 중앙 집중에서 원격으로 이를 소프트웨어로 제어하기 위하여 IPMI (Intelligent Platform Management Interface)[2]와 같은 규격을 지원하는 전원용(Power) 네트워크의 연결이 필요하다.

o Playground 내 SmartX Provisioning Framework를 운용하는 노드인 Provisioning Center 가 존재하며 Playground 운영자는 Provisioning Center와 상호작용 함으로써 Playground 내 분산되어 위치하는 모든 Box에 소프트웨어 설치, 상태 관리 등을 원격/자동으로 제어한다.

o SmartX Provisioning Framework는 OF@KOREN Playground 전체 인프라를 마치 하나처럼 Provisioning 하기 위하여 Playground Template이라는 템플릿을 기반으로 동작한다. 운영자는 Playground를 구성하는 각 Box의 Provisioning 후 상태를 디자인하여 Playground Template에 명시하게 되면, Framework는 해당 템플릿에 명시된 디자인을 기반으로 전체 Playground를 Provisioning 한다.

o 실제 SmartX Provisioning Framework 활용 측면에서 OF@KOREN Playground 내 Box를 대상으로 Ubuntu MAAS(Metal As A Service) 도구[4]와 DevStack 도구[5], 그리고 자체적으로 제작한 OpenStack 설치 스크립트 도구를 활용하여 Ubuntu Linux 및 OpenStack Cloud OS[6]를 자동으로 설치하여 OpenStack 기반의 Cloud Playground를 자동으로 구축하는 것을 하나의 Use Case이자 기능으로써 제공하고 있다.

1.2. SmartX Box

o 본 기술문서의 자동 설치 도구가 대상으로 하는 초융합형(Hyper-convergent) 자원 Box는 ICT 인프라의 기본이 되는 컴퓨팅/네트워킹/스토리지라는 이질적인 자원들을 하나의 Box의 형태로 제공 가능한 Box를 의미한다. 이러한 융합형 자원 박스 개념 이전까지는 상기 자원들을 제공하기 위해서는 특수 목적으로 제작된 하드웨어에 이를 위해 작성된 소프트웨어만을 사용하도록 강요되어 왔다. 하지만 SDN/클라우드/NFV 기술이 각광을 받아 성숙되어 가고 있는 현재 추세와 더불어 x86 기반의 하드웨어 성능이 급격하게 향상되어 컴퓨팅 뿐만 아니라 NFV, 가상화 환경까지 원활하게 지원 가능함이 검증되어 왔다.

o 실제 이러한 추세에 따라 세계적으로 다양한 스타트업기업들이 융합형 자원 박스를 무기로 하여 사업 영역을 확장해 가고 있다. 2013년부터 뉴타닉스(Nutanix), Simplivity, Pluribus 등의 업체들이 컴퓨스토리지 (compu-storage), 서버-스위치 (server-switch) 로 불리는 컴퓨터, 스토리지, 네트워크 등을 결합하는 융합형 인프라 확산을 추구하면서, 용이한 설정, 유연한 운용 지원, 획기적 성능 개선 등을 무기로 scale-in/out 의 장점을 보여주면서 시장 호응을 얻고 있다.
이 뿐만 아니라 Private 클라우드 분야의 강세 기업인 VMware는 Evo:Rail 이라는 융합형 자원 박스의 통합적인 관리를 위한 솔루션을 14년부터 판매하고 있으며, IT 서비스 분야의 거대 기업 페이스북은 자신의 인프라 전체를 자체 제작한 융합형 자원 박스로 대체하였고, 이를 공개하여 오픈소스 융합형 자원 박스 하드웨어 개발을 위한 Open Compute Project를 주도적으로 이끌고 있다.

o 이러한 흐름에 대비하여 국내에서는 GIST의 Networked Computing Systems 연구실의 주도하에 12년~13년에 국내 5개 사이트, 국외 7개 사이트에 한국형 융합 자원 박스로써 SmartX Box를 설계, 배포하여 미래 인터넷 테스트베드의 운용을 시작하였다.[3] 시작 이래로 국내외 연동 사이트 및 Box 수와 같은 규모적 향상 뿐 아니라 초기 SDN, Cloud를 넘어서 IoT, BigData, FastData 등 다양한 연구주제를 위한 실험환경을 제공하는 질적 향상도 꾸준히 도모하고 있다.

o SmartX Box는 클라우드 컴퓨팅의 3대 자원 요소인 컴퓨트/네트워킹/스토리지 자원을 하나의 박스 단위로 확보하여 다양한 서비스에 적합한 자원을 유연하고 신속하게 제공하기 위한 한국형 초융합 자원 박스로 정의한다.

o SmartX Box의 SmartX는 'Smart (지능형)' + 'X (유연하게 변화하여 적응함)'을 연계하는 개념을 대표하는 키워드로써 소프트웨어 중심적인 실증형 기술 개발과 운영을 추구하는 의미를 담고 있다.

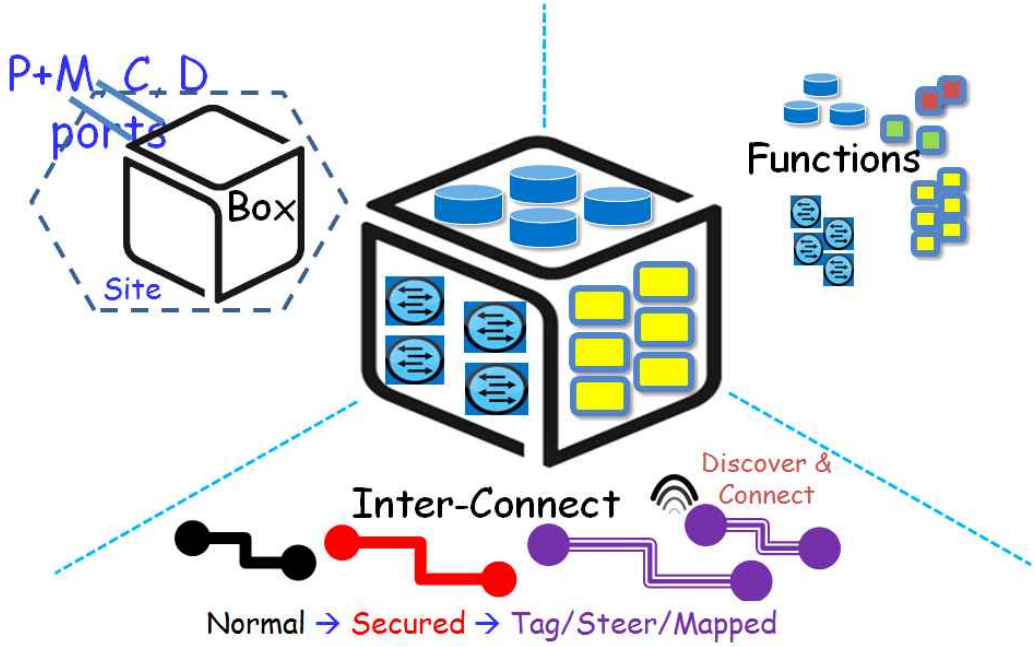

그림 1 융합형 SmartX Box 및 Box/Inter-Connect/Function의 3대 요소

o SmartX Box는 그림 2의 가운데 박스 심볼의 형태로 표현하며 박스 내부의 아이콘은 각각 우측 하단부에 컴퓨팅, 좌측 하단부에 네트워킹 그리고 상단부에 스토리지 자원을 의미한다. 따라서 이와 같이 이질적인 자원들을 하나의 박스에 포함된 그림을 SmartX Box의 심볼로 사용함으로써 SmartX Box가 초융합형 자원 박스임을 상징화 하고 있다.

o 그림 2와 같이 초융합형 Box 자원집합을 기반으로 Box/Inter-Connect/Functions의 3대요소를 개념화해서 이뤄지는 IoT 서비스 개념은 소프트웨어-정의 패러다임에 따라 빅뱅수준의 속도로 발전하는 ICT 인프라/플랫폼/서비스 통합 기술의 발달 차원에서 바라볼 때 단순하고 명료하면서 미래에도 지속 가능한 기술 구도를 제시하고 있다.

o 신속성과 효율성을 무기로 하는 DevOps(개발/운영 병행체제) 방법론을 도입하여 오픈-소스 소프트웨어를 기반으로 소프트웨어-정의 및 가상화를 지원하는 초융합형(컴퓨트/네트워킹/스토리지를 통합한) SmartX Box 형태의 (오픈소스 White Box 하드웨어와 연계된) 자원집합을 요구되는 서비스의 부하에 유연하게 맞추면서 동시에 자원집합 전반에 걸친 활용도를 최적화하여 신속성과 경제성을 동시에 추구하는 개발/운용 자동화 기술이 요구된다.

1.3. SmartX Virtual Playground

o Playground란 사용자들이 하고 싶은 Play를 자유롭게 할 수 있게 제공되는 공간을 상징하고 사용자들이 Playground 위에서 하는 다양한 실증 실험들은 Play로 표현할 수 있다.

o SmartX Playground는 사용자들이 원하는 실증 실험을 할 수 있도록 SmartX Box들을 유연하게 엮어 컴퓨팅, 네트워킹, 스토리지로 대표되는 IT 자원들을 알맞게 제공할 수 있는 물리적인 인프라를 의미한다. SmartX Virtual Playgroud(SmartX 가상놀이터)는 하나의 공통 인프라인 SmartX Playground 상에 DevOps 기반 자동화된 설치/설정 소프트웨어 도구를 활용하여각 사용자의 요구사항에 따라 동적이고 유연하게 제공되는 실증 실험 환경을 의미한다.

o 즉, Playground는 다양한 사용자간 공유되는 물리적인 인프라 이지만 Virtual Playground는 하나의 공유 인프라 상에 각 사용자 별로 독립적으로 제공되는 실증 실험 공간이다. 이러한 Virtual Playground는 사용자의 요구에 따라 자동화된 설치/설정 도구를 통해 동적, 유연, 신속하게 구성되어야 한다.

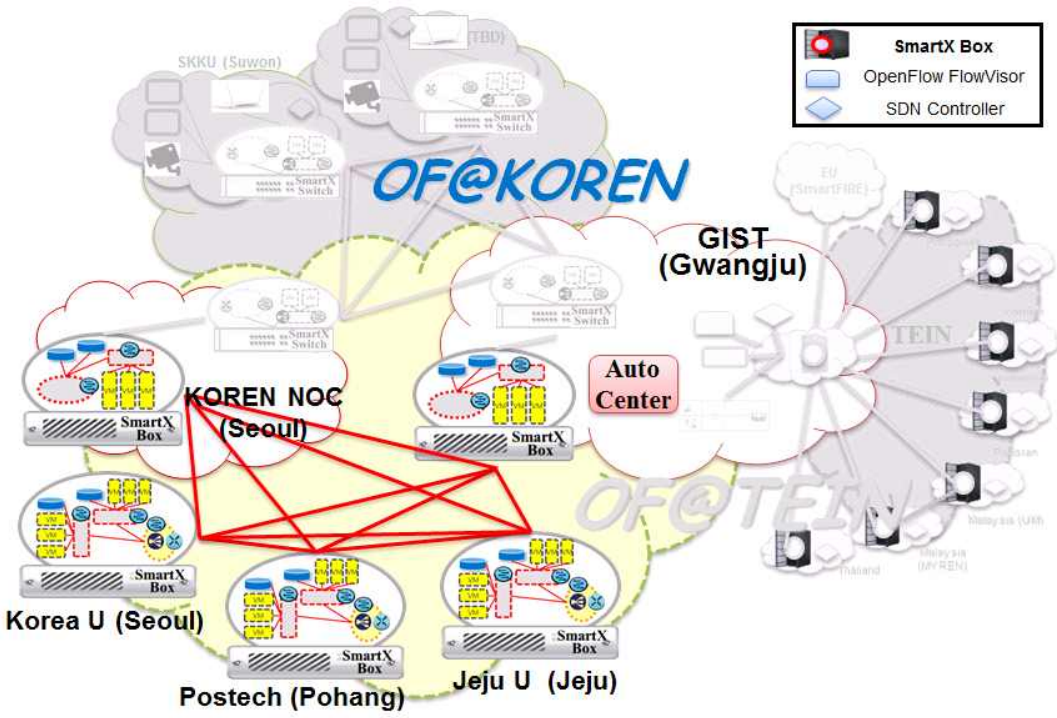

그림 2 OF@KOREN Playground 구성도

o 국내 미래인터넷 관련 연구자들을 대상으로 클라우드 기반의 미래인터넷 실증 실험 테스트베드를 제공하고자 2012년 광주과학기술원 NetCS 연구실을 중심으로

KOREN NOC(Network Operation Center), 고려대학교, 포항공과대학교, 제주대학교를 대상으로 초 융합형 자원 박스인 SmartX Box를 배포하고 이를 KOREN 연구망으로 연결하는 OF@KOREN Playground (i.e. 테스트베드)의 구축을 시작하였다. (그림 2)

o OF@KOREN Playground는 그림 2에서 보는 것과 같이 지리적으로 국내 각지에 분산되어 있는 SmartX Box를 OpenStack으로 엮어 하나의 분산 OpenStack 클라우드로 구성함으로써 컴퓨팅, 네트워킹, 스토리지 자원들을 사용자들에게 제공하고 있다. 이 때 KOREN 연구망을 중심으로 SmartX Box들을 연결하여 구성된 물리적인 인프라는 Playground라고 할 수 있고, 이 인프라 상에 구성된 OpenStack은 각 사용자들에게 독립적인 Cloud 환경인 Virtual Playground를 제공한다.

1.4. 자동 Provisioning의 필요성

o SmartX Playground를 구성하는 SmartX Box들은 시간이 지날수록 그 규모가 증가하고 있다. 2012년부터 시작하여 OpenStack Cloud 기반의 Virtual Playground 환경을 제공하는 OF@KOREN Type C Playground를 그 시초로 하여 광주, KOREN NOC, 경기과학기술진흥원(GSTEP), SMEC, 성균관대에 서버-스위치 형 SmartX Box Type S로 구성된 Type S Playground을 운영하고 있으며, 그리고 향후 POSTECH, 숭실대, 고려대, KAIST, 광운대를 엮는 K-ONE 공동개발환경을 구축할 예정으로 나날이 그 규모가 커지고 있다. 관리해야 하는 박스의 수가 갈수록 증가함에 따라 Provisioning로 소요되는 시간 또한 기하급수 적으로 증가할 수밖에 없다.

o SmartX Playground는 멀티 사이트 환경에서 발생할 수 있는 이슈들을 다루는 것을 목적으로 하기 때문에 데이터센터처럼 하나의 사이트에 박스들이 집중되어 있는 것이 아니라 국내 각지에 분산 배치되어 있는 상태이다. 기존까지는 이와 같은 멀티 사이트 환경에서 운영체제 및 소프트웨어를 설치/설정하기 위해서는 지리적인 제약으로 인하여 시간적, 인적인 자원의 소모가 발생할 수밖에 없었다. 하지만 오늘날 IT 인프라의 중심이 멀티 사이트 규모의 Cloud 환경으로 옮겨 옴에 따라 중앙에서 지리적으로 분산되어 있는 박스들에 운영체제 및 다양한 소프트웨어들을 자동으로 설치/설정해 주는 기술 및 도구들이 개발되고 있는 상황이다.

o SmartX Playground는 기업, 연구소 차원에서 운영하는 인프라 와는 달리 그 목적이 다양한 실증 실험 환경을 제공하는 것에 있다. 따라서 급속도로 발전하는 다양한 소프트웨어와 사용자들의 요구사항에 맞춰 실증환경을 보다 유연하고 동적으로 구성할 수 있는 기능을 갖추어야 한다. 따라서 다양한 소프트웨어의 설치/설정 도구들을 하나의 틀로 엮는 SmartX Provisioning Framework를 개발함으

로써 다양한 소프트웨어의 설치 도구를 확장성 있게 지원해야 한다.

o 따라서 SmartX Playground의 특성인 지속적인 규모 확장, 멀티 사이트 환경, 다양한 실증환경 제공이라는 요구사항에 맞춰 Playground를 효율적으로 Provisioning 하기 위해서는 자동화된 Provisioning Framework를 구축, 운영하는 것이 반드시 필요하다.

2. SmartX Provisioning Framework

2.1. SmartX Provisioning Framework 개요

o SmartX Provisioning Framework 이전부터 SmartX Playground를 운영하는 과정에서 이미 확장성, 멀티사이트 환경을 유연하고 신속하게 제공해야 하는 요구사항을 갖고 있었다. 이러한 이슈를 해결하고자 Ubuntu 리눅스, OpenStack 조합에 대한 자동설치 도구를 개발하여 운영에 활용하였다. [7]

o 기존의 자동 설치도구를 이용하여 Ubuntu, OpenStack의 단순 조합에 Playground 자동 설치/설정은 문제 없이 수행하였다. 하지만 현재 인프라 관련 기술들이 매우 빠르게 발전함에 따라 새로운 소프트웨어의 자동 설치 또한 자동 설치 도구에 손쉽게 추가가 가능해야 함은 물론이고 기존에 구현된 소프트웨어 자동 설치도 손쉽게 수정/보완 가능한 설치 도구들을 하나로 감쌀 수 있는 커다란 틀 역할을 하는 소프트웨어가 요구되었다.

o 따라서 기존에 통합한 Installer 코드의 관리 및 개발을 손쉽게 하는 것뿐만 아니라 새로운 소프트웨어의 Installer 또한 쉽게 통합하고 이러한 Installer를 통합된 방식으로 관리, 운용할 수 있는 SmartX Provisioning Framework의 필요성이 대두되기 시작하였다.

o SmartX Provisioning Framework란 SmartX Playground를 구성하는 멀티 사이트 SmartX Box 들을 대상으로 리눅스, OpenStack 등의 다양한 소프트웨어를 중앙에서 원격으로 자동 설치하고, 이에 수반되는 필요한 설정을 자동화하는 도구들을 하나의 Framework라는 커다란 틀 하에서 엮음으로써 공통된 방식을 통해 다양한 조합의 소프트웨어를 자동으로 설치/설정 할 수 있는 틀을 제공해주는 소프트웨어를 의미한다.

o SmartX Provisioning Framework 자체는 소프트웨어를 자동 설치/설정하는 기능을 제공해 주지 않는다. 각 소프트웨어의 자동 설치 세부 기능은 각각 독립적인 Installer 소프트웨어의 형태로 개발된다. Framework는 이러한 다양한 Installer 들을 엮어 주는 소프트웨어로써 동일한 인터페이스를 통해 다양한 소프트웨어 조합의 설치/설정을 단일화된 로직으로 제어할 수 있는 기능을 제공한다.

o SmartX Provisioning Framework 소프트웨어를 활용함으로써 SmartX Playground 운영자는 다양한 구성의 Playground를 별도의 Installer들을 각각 다룰 필요 없이 하나의 소프트웨어를 통해 효율적으로 Provisioning 할 수 있다.

2.2. SmartX Provisioning Framework 요구사항

o 대규모의 박스들, 멀티 사이트 박스 환경를 대상으로 다양한 서비스 실증 환경을 제공해야 한다는 SmartX Provisioning Framework의 근본적인 목적에 부합하기 위하여 아래와 같은 요구사항을 최대한 만족하도록 Framework를 설계하였다.

o Cloud, BigData, Container 등 현재 IT 분야의 기술 및 소프트웨어들이 엄청난 속도로 발전됨 따라 SmartX Playground 또한 이러한 추세에 맞춰 하나의 고정된 버전의 소프트웨어만을 지원하는 것이 아닌 새로운 소프트웨어에 대한 자동설치/설정을 쉽게 지원해야만 한다. 따라서 새로운 Installer를 Framework에 추가하는 방법을 규정화하고 단순화함으로써 Installer의 개발/개선과 Framework의 개선을 독립적으로 수행할 수 설계한다. 이를 통하여 Framework 기능 측면에서의 확장성, 신속성을 추구한다.

o 아무리 Framework에서 많은 기능을 제공한다고 하더라도 이를 사용하는 운영자가 Framework를 통해 Playground를 원하는 형태로 쉽게 Provisioning 하지 못한다면 운영 도구로써의 의미를 잃게 된다. 따라서 본 Framework를 활용하여 Playground 전반을 소프트웨어로 쉽게 제어하기 위하여 Playground Template 기반의 SmartX Provisioning Framework를 설계한다. 템플릿 기반의 의미는 운영자가 Playground을 어떻게 구성할 것인지를 하나의 Playground Template에 묘사해 놓으면 Framework가 해당 Template를 참고하여 Playground에 필요한 소프트웨어를 자동을 설치/설정해야 함을 의미한다. 따라서 Playground의 운영자는 Provisioning을 위해 Framework에 내포된 Installer 각각을 다룰 필요 없이 희망하는 Playground의 상태를 Template에다가 기재하기만 하면 된다. Template 기반의 Framework 동작 방식은 운영 도구로써의 효율성과 활용 가능성을 크게 높여줄 수 있다.

o Playground란 다양한 실증 환경을 사용자에게 제공할 수 있는 공간이므로 다양한 소프트웨어가 설치되어 운용될 수 있다. 따라서 Playground를 자동으로 구성하는 Provisioning Framework 또한 요구되는 Playground의 형태에 따라 다양한 소프트웨어 Installer들을 순차적으로 엮어서 사용해야 한다. 이 때 순차적으로 설치되는 소프트웨어의 정상 설치여부를 확인하는 기능은 안정성 및 실제 활용가능성 측면에서 SmartX Provisioning Framework에 필수적이다. 또한 이러한 설치 상태 확인 메커니즘은 실시간으로 Box의 상태를 조회하는 기능으로 확장 가능하며, Provisioning 시 이미 설치된 소프트웨어를 대상으로 하는 중복 설치를 예방하는 등 Provisioning Framework 측면에서 많은 효용성을 가져올 수 있는 기능이다.

2.3. SmartX Provisioning Framework 소프트웨어 구현

o 상기 2.2절에서 명시한 요구사항에 따라 SmartX Provisioning Framework의 Prototype을 아래 그림 3과 같은 구성으로 실제 구현하였다.

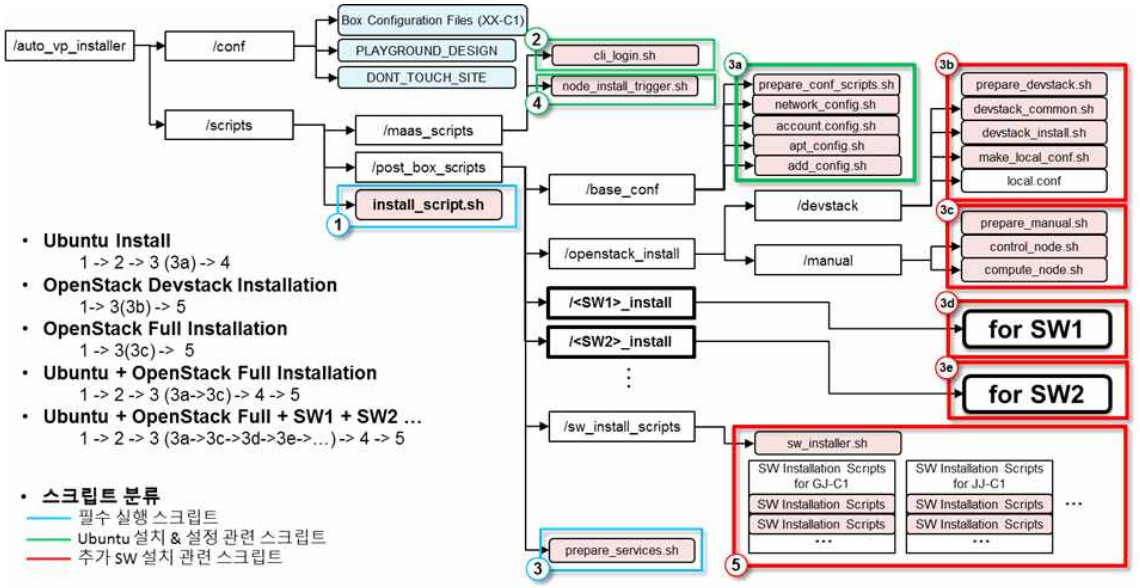

그림 3 SmartX Provisioning Framework 소프트웨어 구조

o SmartX Provisioning Framework의 Provisioning 대상이 되는 Box를 보다 쉽게 추가/제거/관리할 수 있도록 conf 디렉토리 내에 Box의 상세 정보를 각각의 설정파일로 저장해 놓는다. 이 때 각 설정파일에는 Box의 호스트명, Box내 P/M/C/D NIC의 IP 주소, MAC 주소, 기본 계정정보 등이 기재되어 있다. 이 설정파일은 향후 2.4절에서 설명할 Framework 동작 단계 중 "Installation Coordinator에 Box별 Installer Set 생성"에서 Box에 맞는 Script를 생성하는데 사용된다.

o 또한 SmartX Playground의 Playground Template는 conf 디렉토리에 위치한다. (PLAYGROUND_DESIGN) 앞서 설명하였듯 SmartX Playground Operator는 Provisioning Framework를 운용하는데 있어 오직 Template 파일만을 수정하면 된다. Operator가 원하는 Playground의 상태를 정해진 포맷에 맞추어 기재한 후 Framework SW를 실행하면 자동으로 해당 상태에 맞춰 소프트웨어가 설치/설정된다.

o 각 Installer들은 post_box_scripts 디렉토리 내 소프트웨어별로 디렉토리를 구성하여 위치하고 있다. 예를 들어 OpenStack의 설치를 위한 Installer는 openstack_install이라는 디렉토리 밑에 위치하고 있다. 이러한 Installer들은

Provisioning Framework 소프트웨어가 실행되는 과정에서 Playground Template에 맞추어 동적으로 취사선택 된다.

o 현재 Provisioning Framework에 통합되는 Installer 들은 전부 유사한 방식으로 동작하고 있는데, 대상이 되는 Box에 Installation 과정을 제어/관리하는 Installer (현재는 Script)를 전송한 후 이를 Box 내부에서 실행시킴으로써 Installer가 Box 내부를 소프트웨어로 채우는 접근법을 사용하고 있다. 이를 통해 다수의 Box의 설치 과정에서 발생하는 부담을 각 Box로 전이시킴으로써 Provisioning의 규모적인 확장성을 향상시킬 수 있다.

o 새로운 Installer를 추가하는 경우 post_box_scripts 내에 Installer를 위한 디렉토리를 생성한 후 그 안에 Installer 소프트웨어를 구성하면 된다. 이렇게 Installer를 보다 쉽게 통합할 수 있는 구조를 취함으로써 Installer의 개발과 Framework의 개선을 독립적으로 분리할 수 있고, 이를 통해 Framework 관점에서의 확장성을 향상시킬 수 있다.

o Framework 소프트웨어 내에서 실제 Box의 설치를 담당하는 것은 Installation Coordinator 이다. Installation Coordinator는 앞서 Framework의 동작 과정에 따라 취사선택된 Box별 Installer들을 모두 갖고 있으며, Box에 대한 접근 정보 또한 갖고 있다. Installation Coordinator는 Box에 Installer를 배포, 실행하여 실질적인 설치/설정과정을 관리함은 물론이고 나아가 각 소프트웨어의 설치 결과를 조회하여 정상이면 다음 Installer를 실행시키고 문제발생 시 특정 복구지점으로 되돌아가는 등의 보다 지능화된 기능 또한 제공하는 것을 목적으로 한다.

2.4. SmartX Provisioning Framework 동작과정 설명

o SmartX Provisioning Framework의 동작과정은 크게 “Playground Template 작성 및 Framework 실행”, “Playground Template 분석”, “Installation Coordinator에 Box별 Installer Set 생성”, “Installation Coordinator를 이용한 설치”의 4단계로 구분되어진다. 아래 그림 4는 위의 4가지 단계를 시각적으로 도식화 한 것이다.

o 첫 번째 과정인 “Playground Template 작성 및 Framework 실행” 단계에서는 Operator가 본인이 생각하는 Playground의 설정 상태를 Template 파일로 명시한 후 Framework 소프트웨어를 실행하는 단계이다.

o Framework 소프트웨어를 실행시키면 두 번째 단계인 “Playground Template 분석” 단계가 수행된다. Framework 실행 전 작성된 Playground Template을 읽어리고 해석하여 각 Box 마다 설치될 소프트웨어와 사용할 Installer를 확인한다.

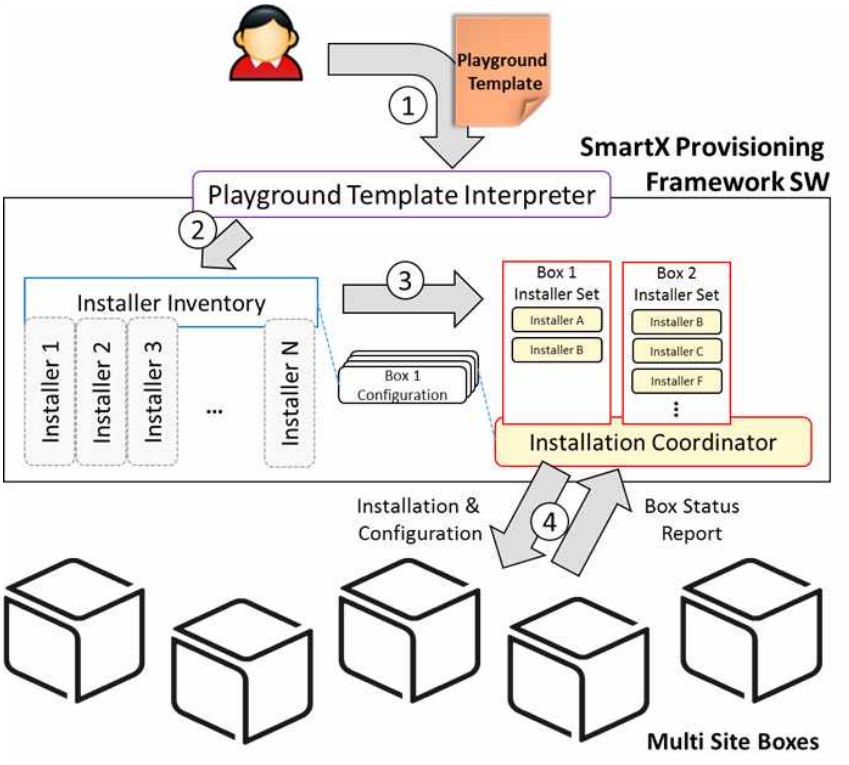

그림 4 SmartX Provisioning Framework Working Procedure

o Framework 소프트웨어 세 번째 단계 “Installation Coordinator에 Box별 Installer Set 생성” 에서는 전 단계에서 Template을 해석하여 확인한 Box별 설치할 소프트웨어 리스트에 맞춰 필요한 Installer들을 Installation Coordinator 내에 Installer Set의 형태로 구성하는 과정이 수행된다.

o 좀 더 자세히 설명하자면 앞에서 언급하였던 것과 같이 현재 구현된 Installer는 특정 소프트웨어를 설치하기 위한 자동화 스크립트의 집합이다. 따라서 각 Installer 그 자체만으로는 설치를 위해 필수적인 Box별 특정 설정값들(IP 주소 등)을 갖고 있지 않기 때문에 이를 실행 단계에서 동적으로 셋팅해 주어야 한다. 따라서 각 Installer 디렉토리 별로 Box를 위한 Installer를 자동으로 생성해주는 스크립트가 존재하여 실행과 동시에 Box 명을 전달하면 Installer 스크립트를 Installation Coordinator 디렉토리에 복사하고 Box 의존 설정값을 설정해 준다. 이 때 미리 작성해 놓은 Box의 설정 파일을 참고하여 Installer의 설정값들을 구성한다.

o 앞에서 반복적으로 설명하였듯 Installation Coordinator 디렉토리로 복사, 수정된 Installer 들을 Box로 복사한 후, Box 내에서 실행시키면 목적 Box 내 목적 소프트웨어의 설치/설정이 자동화 된다. Box마다 하나 이상의 소프트웨어를 설치할

수 있으므로 하나 이상의 Installer가 Box를 위해 준비될 수 있다. 따라서 특정 Box를 위해 복수개의 Installer가 준비되므로 이를 Installer Set이라 한다.

o 여기서 주의할 점은 만일 다른 두 Box가 같은 소프트웨어를 설치/설정한다고 하더라도 Installer 스크립트 내에 설정할 값이 다르므로 같은 소프트웨어 설치를 위해 두 개의 Installer 스크립트가 존재해야 한다는 점이다.

o 마지막 단계인 "Installation Coordinator를 이용한 설치" 단계에서는 Installation Coordinator를 중심으로 실제 원격지에 위치한 Box 내에 소프트웨어를 설치/설정하는 단계이다. 이미 세 번째 단계를 통해 Installation Coordinator가 모든 Box에 필요한 Installer Set을 같은 디렉토리 내에 포함하고 있는 상태이다. 따라서 앞서 여러번 설명하였듯 Installation Coordinator는 이미 준비된 Installer Set에서 순차적으로 Installer를 Box로 복사한 후 이를 실행시키는 방식으로 Box에 소프트웨어를 순차적으로 자동 설치/설정한다.

o Installation Coordinator는 항상 소프트웨어의 설치시작 전, 대상 Box의 상태정보 및 동작 중인 소프트웨어 리스트를 확인한다. 이러한 확인 과정을 통해 동일한 소프트웨어의 중복 설치를 방지할 수 있고, 직전에 설치한 소프트웨어의 설치 결과를 조회하여 문제 발생 시 이를 재설치하거나 특정 복원 지점으로 복구하는 메커니즘을 적용할 수 있기 때문이다.

o 현재 구현된 Framework에서는 상태 및 SW 리스트를 조회하여 이전 소프트웨어 설치가 완전히 완료된 이후에서야 그 다음 소프트웨어의 설치를 시작한다. 또한 소프트웨어 설치에 문제가 발생했을 경우 단순히 에러 메시지를 출력하는 수준으로만 동작하고 있다.

o Installation Coordinator는 소프트웨어 설치 시 모든 Box를 대상으로 설치를 병렬적으로 수행한다. 만일 4개의 Box를 대상으로 설치/설정을 진행할 경우, Box를 순차적으로 설치하는 것이 아니라 4개 Box 설치를 동시에 병렬적으로 진행함으로써 설치에 소요되는 시간을 단축시켜 신속한 설치를 가능케 한다.

o Installation Coordinator가 모든 Box에 대하여 병렬/반복적으로 Box 상태 확인, Installer 복사, 실행을 함으로써 모든 소프트웨어를 설치/설정하게 되고 결과적으로 SmartX Playground 전체 관점에서의 Provisioning을 자동화하게 된다.

2.5. SmartX Provisioning Framework SW 설정 및 실행방법

o 현재 구현된 자동 설치 도구는 /home/tein/auto_vp_installer 에 위치하는 것을 가정하여 동작한다. 다른 디렉토리에 구축하고자 하는 경우, 각 스크립트 파일 내에 "INSTALLER_PATH"를 자동 설치 도구의 위치에 맞춰 수정해 주어야 한다.

o 계정 sudo 권한 부여
자동 설치 도구는 sudo 권한을 요구하므로 편의를 위하여 다음 커맨드를 입력하여 sudoers 파일에 권한을 등록하는 것을 추천한다.

$ sudo -c 'echo "<계정명> ALL=(ALL:ALL) NOPASSWD: ALL">>/etc/sudoers'

o 노드 구성 파일 생성
원격 자동 설치/설정 도구를 활용하여 특정 박스를 설치/설정 하기 위해서는 해당 박스의 네트워크 인터페이스 정보가 필요하다. 이 정보를 기반으로 Ubuntu 설치 이후 네트워크 인터페이스 설정, Installer 스크립트 설정값 변경, DevStack 설정 파일 자동 생성 등 특정 박스에 맞는 설정을 자동화 하게 된다. 따라서 특정 박스를 자동 설치하기 위해서는 "<도구 경로>/conf/<호스트명>" 의 경로에 아래와 같은 서식에 맞춰 설정 파일을 생성해 주어야 한다.

만일 박스 호스트 명이 GJ-C1인 경우 "<도구 경로>/conf/GJ-C1" 으로 파일을 생성해야 한다.

```
HOSTNAME=<호스트명>

MGMT_GATEWAY=<게이트웨이IP:-210.114.90.1>
MGMT_DNS_NAMESERVER=8.8.8.8

MGMT_IP_ADDRESS=<관리네트웍 IP>
MGMT_IP_SUBNET=<관리네트웍 서브넷>
MGMT_MAC_ADDRESS=<관리네트웍 IF MAC>

DATA_IP_ADDRESS=<정보네트웍 IP>
DATA_IP_SUBNET=<정보네트웍 서브넷>
DATA_MAC_ADDRESS=<정보네트웍 IF MAC>

CTRL_IP_ADDRESS=<제어네트웍 IP>
CTRL_IP_SUBNET=<제어네트웍 서브넷>
CTRL_MAC_ADDRESS=<제어네트웍 IF MAC>
```

그림 5 Framework 내 박스 설정 파일 규격

o SmartX Playground Template 작성법
원격 자동 설치/설정 도구는 도구 내에 정의된 템플릿 파일을 기준으로 도구에

등록된 융합형 자원 박스들을 쉽고 유연하게 설치/설정하게 된다. 따라서 도구를 통해 원하는 환경으로 설치/설정하기 위해서는 자체적으로 규격화한 템플릿 서식에 따라 원하는 설치/설정 환경을 반드시 정의해야 한다.
템플릿은 도구 내 "<도구 경로>/conf/PLAYGROUND_DESIGN" 파일이다. 아래 그림은 기본적인 템플릿 규격이다.

```
TARGET_NODES: <설치 박스명>
<박스명> {
        INSTALL_SOFTWARE: Ubuntu
}
<박스명> {
        INSTALL_SOFTWARE: Ubuntu Openstack
        OPENSTACK_MODE: <Controller | Compute>
        OPENSTACK_INSTALLER: <Devstack | Full >
}
```

그림 6 Playground Template 기본 포맷

템플릿 작성의 이해를 돕기 위해 아래 2가지 디자인과 이에 따른 템플릿을 제시한다.

```
TARGET_NODES: GJ-C1 JJ-C1 PO-C1 KU-C1 KN-C1

GJ-C1 {
        INSTALL_SOFTWARE: Ubuntu Openstack
        OPENSTACK_MODE: Controller
        OPENSTACK_INSTALLER: Full
}
JJ-C1 {
        INSTALL_SOFTWARE: Ubuntu Openstack
        OPENSTACK_MODE:  Controller
        OPENSTACK_INSTALLER: Full
}
PO-C1 {
        INSTALL_SOFTWARE: Ubuntu Openstack
        OPENSTACK_MODE:  Controller
        OPENSTACK_INSTALLER: Full
}
KU-C1 {
        INSTALL_SOFTWARE: Ubuntu Openstack
        OPENSTACK_MODE: Controller
        OPENSTACK_INSTALLER: Full
}
KN-C1 {
        INSTALL_SOFTWARE: Ubuntu Openstack
        OPENSTACK_MODE:  Controller
}
```

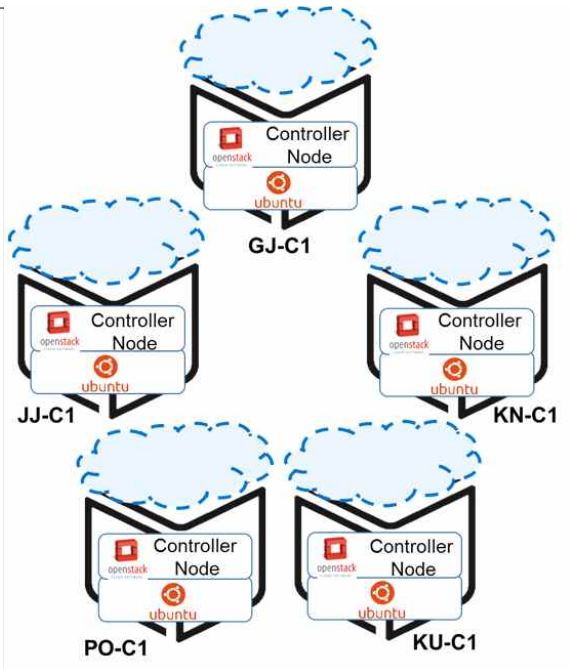

그림 7 Playground Template 작성 예시: 5개 개별 클라우드 환경

```
TARGET_NODES: GJ-C1 JJ-C1 PO-C1 KU-C1 KN-C1

GJ-C1 {
        INSTALL_SOFTWARE: Ubuntu Openstack
        OPENSTACK_MODE: Controller
        OPENSTACK_INSTALLER: Full

}
JJ-C1 {
        INSTALL_SOFTWARE: Ubuntu Openstack
        OPENSTACK_MODE:  Compute
        CONTROLLER_NODE: GJ-C1
        OPENSTACK_INSTALLER: Full

}
PO-C1 {
        INSTALL_SOFTWARE: Ubuntu Openstack
        OPENSTACK_MODE:  Controller
        OPENSTACK_INSTALLER: Full

}
KU-C1 {
        INSTALL_SOFTWARE: Ubuntu Openstack
        OPENSTACK_MODE: Compute
        CONTROLLER_NODE: PO-C1
        OPENSTACK_INSTALLER: Full

}
KN-C1 {
        INSTALL_SOFTWARE: Ubuntu
}
```

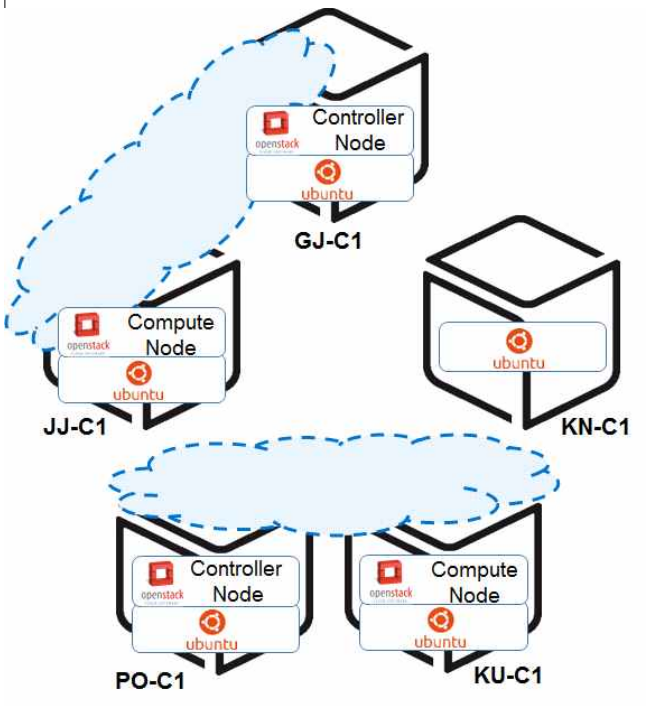

그림 8 Playground Template 작성 예시: 2개 멀티노드 클라우드, 1개 Ubuntu

o Framework를 이용하여 원하지 않은 Box에 자동 설치/설정을 할 경우 현재 운용 중인 환경에 크게 악영향을 미칠 수 있으므로 일차적으로 실수로 인한 자동 설치를 방지하기 위하여 conf/DONT_TOUCH_SITE 파일을 활용한다. 이 파일에 설치를 금지할 박스 명을 기재하면 도구를 실행시키더라도 해당 박스의 설치를 진행하지 않는다.

o 상기 과정을 따라 Framework 소프트웨어를 정상적으로 설정한 후 원하는 설정에 맞춰 템플릿을 작성하였다면 도구를 이용하여 실제 융합형 자원 박스들을 원격 자동 설치/설정하는 것이 가능하다.
Framework를 실행하는 것은 간단하다.

$ bash ./<도구 경로>/scripts/start_init.sh

o 새로운 도구를 추가하기 위해서는 “<FW PATH>/auto_vp_installer/scripts/post_install_scripts/” 디렉토리 밑에 새로운 디렉토리를 생성한 후 installer 스크립트를 배치한다.

그리고 Installer 를 Installation Coordinator 디렉토리에 복사하고 Box에 맞는 설

정값을 수정해주는 스크립트를 작성해 주어야 한다.
예를 들어 Playground Template 내에 Box A에 OpenStack을 설치해야할 소프트웨어라고 명시한 경우, Framework는 자동으로 OpenStack 설치용 Installer 스크립트를 Installation Coordinator로 복사하고 스크립트 내 설정값을 Box에 맞게 수정해 주어야 한다.
현재 구현된 Installer에서는 Installer 디렉토리 내 "prepare_XX.sh" 라는 이름의 스크립트가 이러한 역할을 수행한다.

마지막으로 Playground Template 내에 특정 소프트웨어의 설치가 명시되었을 때 앞에서 작성하였던 Installer 스크립트를 Installation Coordinator 디렉토리로 복사해주는 스크립트를 자동으로 실행해주는 부분을 추가해 주어야 한다. 즉, 바로 전 단계에서 작성한 스크립트(Installation Coordinator 디렉토리로 복사, 설정값 수정)를 Template 분석 결과에 따라 자동으로 불러내는 연결고리를 만들어 주어야 한다. 이러한 연결고리를 수행하는 것이 post_box_scripts 디렉토리 내 prepare_services.sh 라는 스크립트이다.
아래의 그림 내 예시를 참조하여 새로운 Installer 추가 시 스크립트 내에 코드를 추가해 주면 Template의 Parsing 결과에 따라 자동으로 이를 호출하게 되어 Framework 동작 3단계를 자동으로 수행하게 된다.

```
SERVICE_CHECK=‘echo $SERVICES | grep -i <소프트웨어명>‘

if [ "${SERVICE_CHECK:-null}" != null ]; then
        echo "*** You chose <소프트웨어명> as the installed software"
        bash $POSTSCRIPTS_DIR/<INSTALLER DIR>/prepare_XX.sh $TARGET
fi
```

그림 9 새로운 Installer 추가시 prepare_services.sh에 추가할 코드 양식

3. SmartX Playground를 위한 Framework 활용사례 및 설정방법

3.1. SmartX Provisioning Framework의 활용을 위한 Use Case

o SmartX Provisioning Framework은 멀티사이트 환경의 SmartX Playground 인프라를 소프트웨어 도구를 활용하여 자유자재로 설치/설정/제어하고자 하는 Software-defined Infrastructure의 개념을 SmartX Playground 상에 구현하고 실제 운영에 적극 활용하는 것을 목적으로 한다. 따라서 Framework 소프트웨어의 설계 및 구현에서 끝난 것이 아니라 이미 오픈스택 기반의 Cloud Playground 라는 Use Case에 맞춰 Installer를 구현하였으며 이를 Framework에 통합하여 SmartX Playground를 자동으로 Provisioning 하는데 활용해 오고 있다.

o 멀티사이트 Box를 대상으로 오픈스택 기반의 Cloud Playground를 구축하기 위해서는 각각의 Bare-metal Box에 리눅스 운영체제를 설치하고, 리눅스 운영체제가 설치된 Box들 상에 오픈스택의 패키지를 정확하게 설치/설정해야 하나의 Cloud Playground를 구축할 수 있다.

o 하지만 원격지에 분산되어 있는 멀티사이트 기반의 인프라 특성을 고려하였을 때 지리적으로 원격에 위치한 Bare-metal Box에 리눅스를 설치하기 위해서는 로컬에 위치한 PC나 서버에 CD, USB 등을 활용하여 리눅스를 설치하는 방식은 사실상 불가능하다.

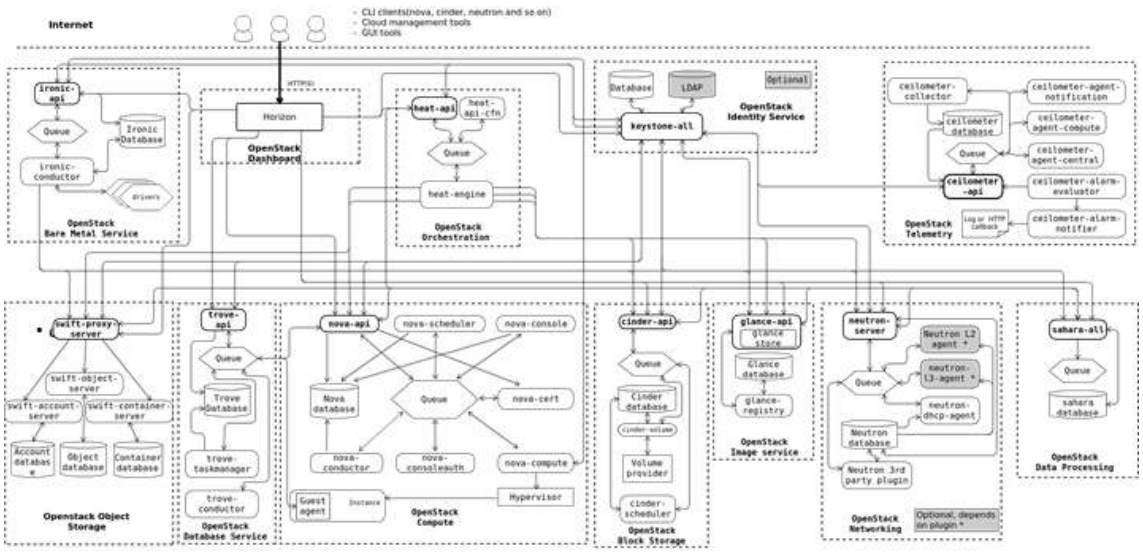

그림 10 오픈스택 클라우드의 논리적 아키텍쳐

o 그림 10을 보면 알 수 있듯 오픈스택이라는 클라우드 운영체제는 복수의 프로젝트가 긴밀하게 얽혀져 동작하는 매우 거대한 소프트웨어이다. 따라서 오픈스택을

어떻게 설치/설정하느냐에 따라 매우 다양한 형태로 구성/운영할 수 있는 장점을 갖고 있다. 하지만 반면에 이러한 복잡성으로 인해 제대로 된 오픈스택 기반 Cloud 환경을 구축하기 위해서는 다수의 Box를 대상으로 복잡한 설치/설정을 실수 없이 정확히 수행해야만 비로소 하나의 Cloud Playground로써 기능할 수 있게 된다. 보통 리눅스가 설치되어 있는 하나의 Box 상에 수동으로 오픈스택을 설치하는 경우에도 대략 1시간 이상의 시간이 소요되는 것을 가정하였을 때 총 5개의 Box들이 얽혀 오픈스택 클라우드를 구성하는 SmartX Playground 환경에서 이를 수동으로 설치하였을 때 매우 많은 인적, 시간적인 자원이 소요됨을 유추할 수 있다. 게다가 5대의 Box가 모두 하나의 사이트가 아닌 지리적으로 분산되어 있는 멀티 사이트 환경에서는 소요되는 노력과 시간이 더욱 증가할 수 밖에 없다.

o 이러한 문제들은 당연하게도 많은 IT 관련 전문가들에 의해 끊임없이 제기되었고 이를 해소하기 위한하여 DevOps 방법론을 기반으로 대량의 융합형 박스들을 중앙에서 쉽고 빠르게 설치하기 위한 다양한 목적을 갖는 자동 설치/설정 도구들 또한 개발되어 왔다. 이러한 도구에는 Bare-metal 융합형 Box 상에 Linux OS(Operating System)을 자동 설치하는 Bare-metal Installation 도구 (또는 Linux Installation 도구)와 Linux 상에 원하는 소프트웨어를 자동으로 설치하는 범용 Software Installation 도구가 있다. Bare-metal Installation 도구에는 Cobbler, OpenQRM, Ubuntu MAAS, Spacewalk 등이 해당하며, 범용 SW Installation 도구에는 Ansible, Puppet Lab, Chef, Ubuntu Juju 등이 이에 해당한다.

o 본 기술 문서에서는 이러한 많은 DevOps 도구들 중 SmartX Playground에 가장 적합하다 생각되는 도구들을 선정하여 이를 Framework가 활용가능한 Installer로 사용하고 있다. Bare-metal 상태의 융합형 Box 상에 Ubuntu Linux를 설치하기 위하여 Ubuntu MAAS[2]를 활용하며 이어서 DevStack[7]을 활용하여 복수의 Box에 OpenStack 클라우드 OS를 설치하여 클라우드 인프라를 자동으로 구성하는 식으로 운영하고 있다. 근래에는 DevStack이 오픈스택 개발자를 위한 설치 도구라는 특성에서 기인하는 운영적인 한계점을 개선하기 위하여 Ubuntu 리눅스 상에 오픈스택을 자동으로 설치하기 위한 스크립트 기반의 Installer를 자체적으로 개발하여 Framework에 통합하여 사용하고 있다.

o 따라서 Ubuntu MAAS를 이용한 Ubuntu 설치, DevStack 활용한 OpenStack 설치, 스크립트 기반 Full 버전 오픈스택 설치 각각을 위해 작성된 Installer에 대해 상세히 설명하고 이를 Framework에 통합하여 활용하기 위한 설정 방법에 대해 설명한다. 또한 Ubuntu Installer와 DevStack Installer를 활용하여 자동 설치/설정하는 Use Case를 대상으로 설치 전체 과정 및 소요 시간을 제시한다.

3.2. Ubuntu MAAS 개요 및 설정

3.2.1. Ubuntu MAAS 개요

o Ubuntu MAAS는 Ubuntu Linux Distribution을 개발/관리하는 기업인 Canonical 사에서 Ubuntu Linux의 원격 자동 설치를 위해 개발한 Bare metal Installation 도구이다. 이 때 MAAS는 Metal As A Service로써 Bare-metal의 metal을 의미하여 Bare-metal 박스의 Ubuntu 자동 설치를 일종의 서비스로 제공하겠다는 의미이다.

o Canonical에서 만든 Bare-metal Installation 도구로써 초기에는 Ubuntu의 자동 설치만을 지원하는 한계가 있었다. 하지만 MAAS 1.7 이후의 버전부터 CentOS, Windows, OpenSUSE와 같은 타 운영체제의 설치를 지원하여 Bare-metal Installation의 대표적인 도구로 발전시키고자 하는 움직임을 보이고 있다.

o Ubuntu MAAS는 "쉽게 구축하고 쉽게 사용가능하며" 분산되어 있는 다수의 서버를 대상으로 "확장 가능하고" "빠르게 Ubuntu를 설치" 가능한 도구를 목표로 개발되고 있다.

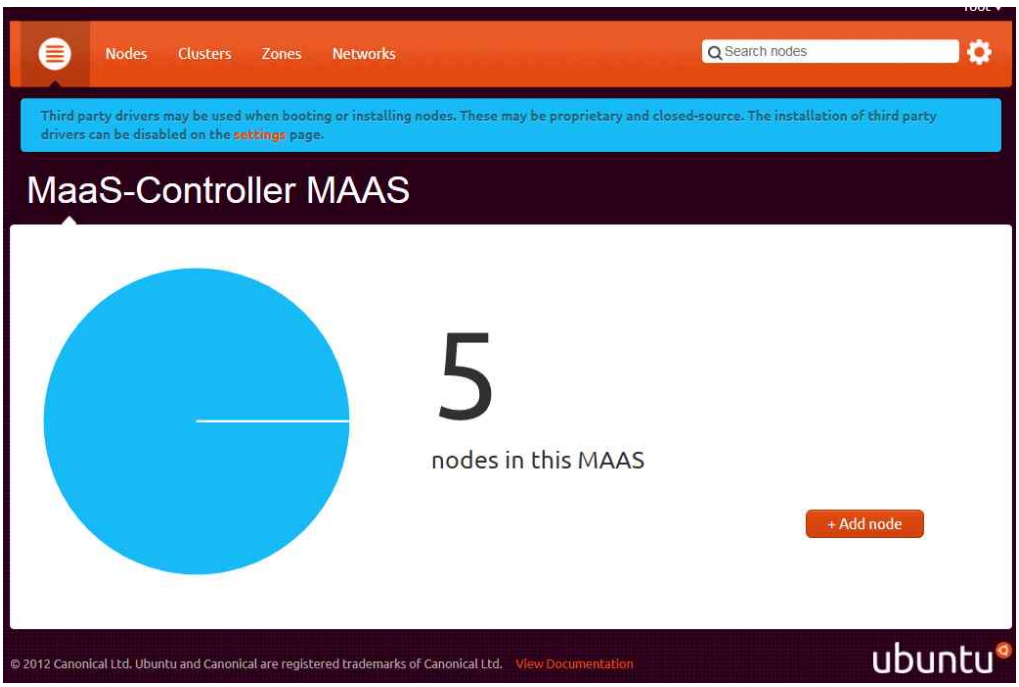

그림 11 Ubuntu MAAS Web UI

커맨드 한 줄이면 MAAS의 설치가 완료되고 그림과 같이 기본 제공되는 Web UI를 통해 클릭 몇 번으로 설치 이미지 저장, DHCP 설정, 노드 전원 설정 등을 모두 설정 가능하다. 또한 한번 설정해 놓으면 설치 이미지를 자동으로 최신으로

유지하는 기능을 제공한다.

또한 그림 12에서 나타나듯이 내부 구조를 Region/Cluster Controller로 계층화하고, Cluster Controller를 복수 배치 가능케 함으로써 서버 수, 시스템 구성에 따라 Operator가 손쉽게 그 규모를 확장 가능하다.

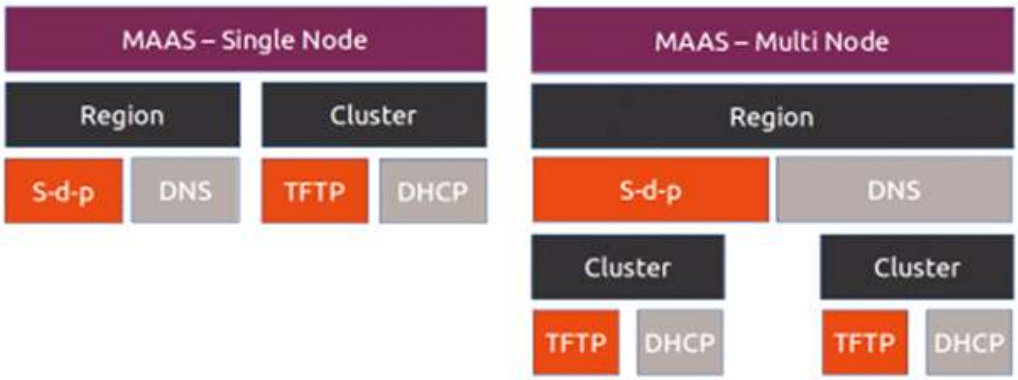

그림 12 Ubuntu MAAS Architecture

Ubuntu MAAS가 Canonical에서 제작된 툴이기에 Ubuntu의 설치를 최단 시간 안에 수행 가능케 하는 Fast-path installer (Curtin)를 기본적으로 제공한다. 다만 아직까지 개발 중인 단계의 Installer이며 커스터마이징 하기 위한 도구 및 매뉴얼 등을 제공하지 않아 활용에 제한적이지만 Ubuntu 설치 시간을 절반 이하로 감소시켜 주기 때문에 향후 대량의 노드를 신속하게 구성해야 하는 IoT 등의 환경에서 그 효용성이 극대화 될 것으로 예측된다.

그림 13 Ubuntu MAAS Working Procedure

o Ubuntu MAAS가 동작 원리는 크게 보았을 때 그림 13와 같이 나타낼 수 있다.

- Enlistment: 새로운 박스를 MAAS에 등록하는 과정
- Commissioning: 등록한 박스의 하드웨어 정보(CPU 코어, RAM, 디스크 사이즈 등)을 수집하는 과정
- Deployment: Commissioning이 완료된 박스에 Ubuntu를 설치하는 과정

o 결과적으로 Ubuntu MAAS는 상기 단계에 따라 복수 노드의 Ubuntu의 설치를 자동화 하는 역할을 수행하는 Bare-metal Installation 도구라고 정의 가능하다.

3.2.2. Ubuntu MAAS 설정

o 본 설정 방법은 Ubuntu MAAS가 SmartX Provisioning Framework 소프트웨어가 동작하는 동일한 Box 내에 설치됨을 가정한다. 향후 개선을 통해 Framework와 Ubuntu MAAS를 독립적으로 분리할 계획이다.

o Ubuntu MAAS를 설치하는 방법은 크게 2가지 방법이 존재한다.

- Ubuntu를 설치 시 Ubuntu MAAS를 동시 설치

Bare-metal 상태의 Box에 Ubuntu를 새로 설치하는 경우 아래 그림과 같이 설치 옵션 중의 하나로 "Multiple server install with MAAS"을 선택한 후 과정에 따라 설치를 완료하면 Ubuntu MAAS가 포함된 Ubuntu를 설치할 수 있다.

그림 14 Ubuntu CD를 통한 Ubuntu MAAS 설치

- Apt-get을 이용한 설치

만일 Ubuntu가 이미 설치되어 실행 중이라면 아래 커맨드를 이용하여 간단하게 Ubuntu MAAS를 설치할 수 있다.

$ sudo apt-get install maas

o 설치 후 Ubuntu MAAS 기본 설정

- Admin 계정 생성

Ubuntu MAAS를 사용하기 위해서는 Admin 계정을 기본적으로 생성해야 한

다. 아래 커맨드를 이용하여 Admin 계정 및 패스워드를 설정한다.

$ sudo maas createsuperuser

- Ubuntu MAAS 웹 페이지 접속

설치된 MAAS의 Web GUI는 http://<IP address>/maas 로 접근 가능하다. 전 단계에서 생성한 Admin ID와 PW로 로그인이 가능하다.

- Cluster Controller의 Image Import

Web GUI에 로그인 후 Clusters 탭을 클릭하면 아래 그림과 같은 화면이 나타난다. Ubuntu MAAS 에서는 노드를 Enlist, Commissioning, Install 하기 위한 커널 이미지를 Cluster Controller에 저장한 후 각 과정마다 알맞은 이미지를 노드에 전송하고 로드하게 된다. 따라서 처음 Cluster Controller 생성시 Import boot images 버튼을 클릭하여 Cluster Controller에 이미지들을 Import 해야 한다. 이때 Ubuntu MAAS 1.5 버전 기준으로 진행 상황이 표시되지 않으며, 다만 GUI 상단의 오류 메시지로 확인이 가능하다. Import에는 약 20분이 소요된다.

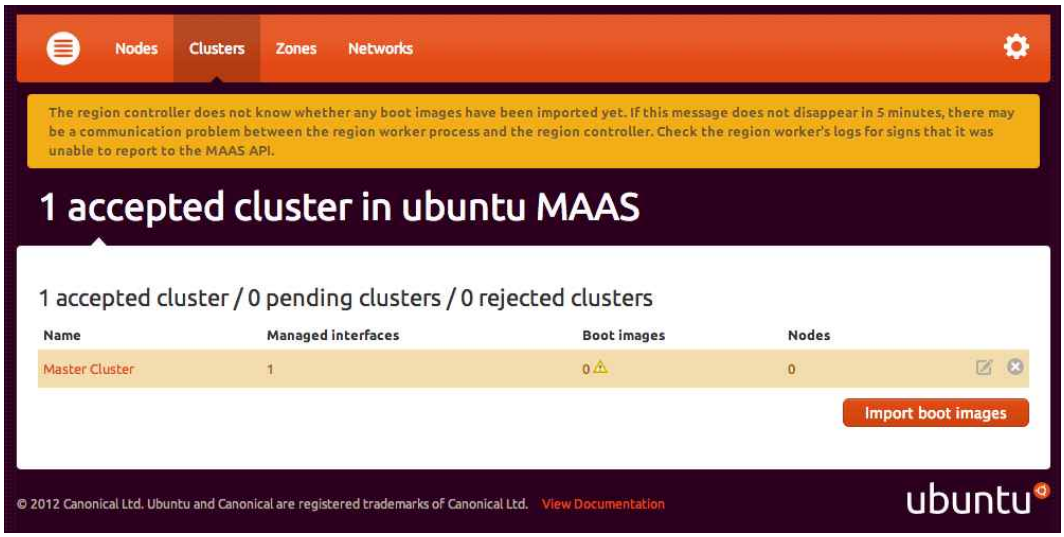

그림 15 Ubuntu MAAS Cluster 탭 화면

- Cluster Controller의 DHCP 설정

Cluster Controller의 이름을 클릭한 후 나타나는 화면에서 “Add interface” 또는 이미 원하는 인터페이스가 있는 경우 우측의 연필 아이콘을 눌러 인터페이스 설정 화면으로 이동 가능하다.

해당 Interface의 Management를 “Manage DHCP and DNS” 로 변경한 후 아래 항목을 채워준다. 입력한 정보를 기준으로 DHCP 서버가 설정되어 구동되며, 이 인터페이스로 접근이 가능한 PXE Booting이 Enable 된 모든 노드들은 부팅시 자동으로 해당 Cluster Controller에 Enlistment 과정을 수행하게 된다.

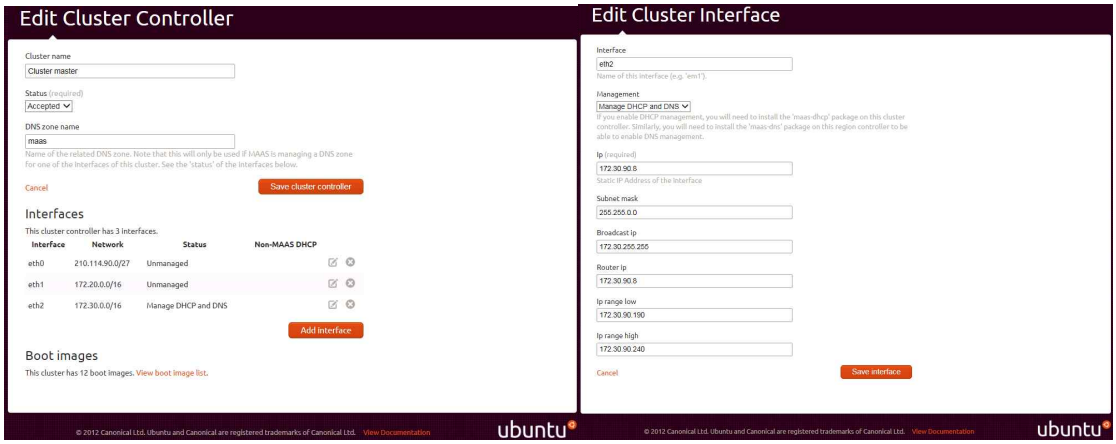

그림 16 Ubuntu MAAS 인터페이스 DHCP 설정

- SSH Key 등록

Ubuntu MAAS를 이용하여 박스의 운영체제를 설치하는 과정에서 MAAS 는 등록한 SSH 공개키를 호스트에 주입하여 해당 SSH 키의 개인키 쌍을 갖는 계정만 SSH로 접근이 가능하도록 제한하고 있다. 따라서 MAAS로 설치한 박스에 SSH로 접속을 원하는 호스트는 SSH 공개키를 등록해야 한다.

$ ssh-keygen

위의 커맨드로 키를 생성하면 계정의 Home directory 밑 .ssh 디렉토리에 ssh 공개키/개인키가 생성된다.

그리고 아래 그림과 같이 MAAS Web UI에 접속하여 우측 상단의 <계정>-Preferences를 클릭한 후, “Add SSH Key”를 클릭하고 Public Key 란에 생성된 공개키인 ~/.ssh/id_rsa.pub의 내용을 복사하여 아래 그림과 같이 추가한다.

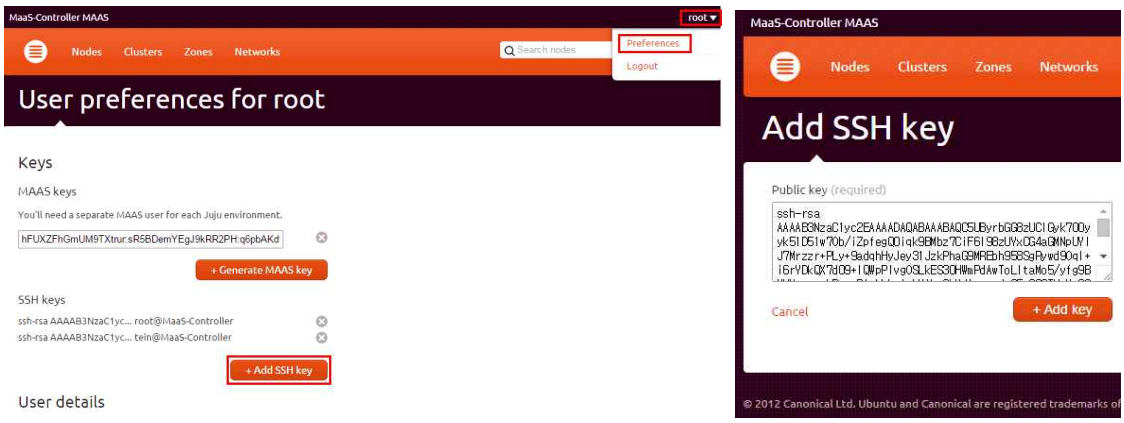

그림 17 Ubuntu MAAS SSH 키 등록

3.2.3. 융합형 자원 박스 전원관리 설정

o 박스가 Ubuntu MAAS와 상호작용하기 위해서는 박스의 인터페이스의 PXE 부팅 [8]을 반드시 enable 시켜주어야 하며 PXE 부팅의 부팅 우선순위를 최우선으로 변경해야 한다. 그래야만 박스가 재부팅 되는 과정에서 PXE 부팅을 통해 Ubuntu MAAS와 연결되어 Enlistment, Commissioning 등 Ubuntu MAAS의 필수 단계가 정상적으로 수행되어 Ubuntu MAAS를 이용해 자동 설치가 가능해진다. PXE 부팅과 부팅 우선 순위 설정은 박스 부팅 시 BIOS 설정에서 변경 가능하다.

o 다수의 융합형 박스들을 중앙 집중적으로 제어하기 위해서는 Ubuntu MAAS가 전원용 네트워크를 통해 각 박스들의 전원을 제어할 수 있도록 설정하는 것이 필요하다.

o 각 서버 벤더들은 자사 원격 관리 솔루션을 판매/제공하고 있다. 현재 설치를 대상으로 하고 있는 OF@KOREN 테스트베드 SmartX Box의 경우 인텔의 ONP 서버를 기반으로 하고 있어 인텔의 RMM(Remote Management Module)을 통해 IPMI로 원격 시스템/전원 관리가 가능하다. 그러므로 인텔의 RMM을 기준으로 설명을 진행한다.

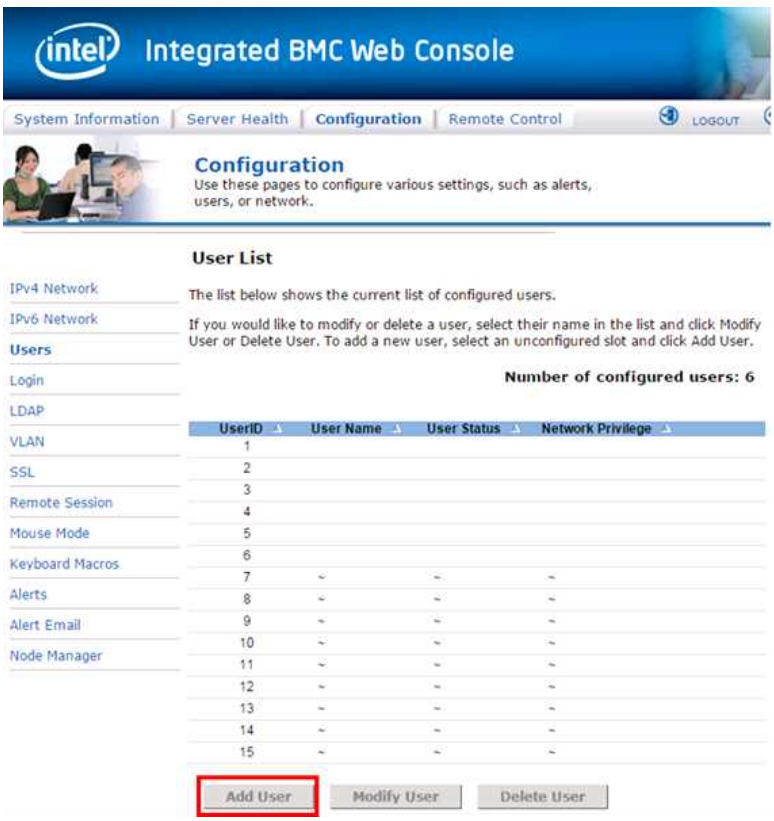

그림 18 Intel RMM 계정 등록

- Intel RMM 설정

Ubuntu MAAS에서 박스를 원격에서 제어하기 위해서는 RMM을 제어하기 위한계정을 생성해야 한다.

박스의 RMM에 접속하여 Configuration 탭으로 이동한 후, 위 그림과 같이 Users에서 Add User를 클릭하여 새로운 계정을 생성한다.

이 때 생성하는 계정은 Operator 또는 Administrator의 권한을 가져야만 MAAS가 해당 계정을 통해 박스의 전원을 제어 가능하다.

- Ubuntu MAAS 노드 설정

Cluster Controller의 DHCP 셋팅이 정상적이고, 박스의 인터페이스 또한 PXE 부팅이 Enable 되어 있다면 재부팅 시 자동으로 MAAS 서버에 Enlistment 되어 MAAS Web GUI의 Nodes 탭에서 확인 가능하다.
그림 19와 같이 Enlist 된 노드를 선택한 후 Edit Node를 클릭하여 노드 설정 화면으로 들어간다.

Power Type을 IPMI, Power Driver를 IPMI 2.0으로 셋팅한 후 아래 RMM의 IP와 이전에 RMM에 등록한 계정의 ID/PW를 입력한다.

o 하지만 Ubuntu MAAS의 경우 IPMI 뿐 아니라 WOL (Wake-On-LAN), HP iLO, Intel AMT 등 다양한 규격을 지원하므로 융합형 자원 박스가 IPMI를 지원하지 않는 환경에서도 Ubuntu MAAS를 활용한 원격 자동 설치/설정이 가능하다.

o 상기 설정을 마치면 Ubuntu MAAS Web GUI를 통해 단순히 대상 박스의 Ubuntu 자동 설치/제거까지 손쉽게 수행 가능하다. 그러나 이 기술문서에서 기술하는 융합형 자원박스 자동 설치 도구를 MAAS와 연동시키기 위해서는 추가적인 설정이 필요하다.

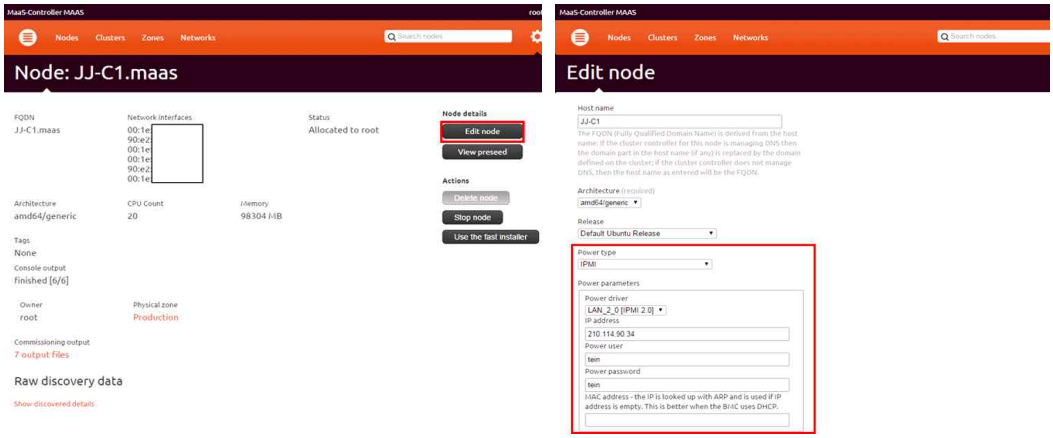

그림 19 Ubuntu MAAS 노드 Power 설정

3.2.4. 원격 자동설치 도구 준비 및 설정

o MAAS DHCP 설정

Ubuntu MAAS를 이용하여 자동 설치 시 각 박스들은 PXE 부팅 과정에서 인터페이스에 셋팅된 DHCP IP 범위 중 하나의 IP를 동적으로 할당 받게 된다. 하지만 고정적으로 운용되어야 하는 테스트베드와 같은 환경에서 DHCP를 통한 IP 동적 할당은 박스 사용의 어려움을 배가시키는 요소이다. 따라서 박스에 고정된 IP를 설정하는 것이 필수적이나 MAAS에서는 기본적으로 이를 위한 기능을 제공하지 않는다. 그러므로 수동으로 DHCP 설정을 변경해 주어야 한다.

- dhcp 설정 파일 수정

/etc/maas/dhcpd.conf 파일을 열고 그림 20과 같이 파일 마지막 부분에 각각 박스에 대한 설정을 추가한다.

각 호스트의 설정을 파일에 기재함으로써 각 박스가 PXE 부팅 시 기재된 MAC 주소로 DHCP request를 MAAS DHCP 서버에 전송하면 fixed-address 항목에 기재한 IP를 할당해 줌으로써 항상 지정된 IP를 할당 받을 수 있다.

```
group {
        host <호스트명>{
                hardware ethernet <PXE IF MAC ADDR>;
                fixed-address <PXE IF IP ADDR>;
                option host-name "<호스트명>";
                option subnet-mask <SUBNET MASK>;
                if option arch = 00:07 {
        filename "bootx64.efi";
                } elsif option arch = 00:0C {
                        filename "bootppc64.bin";
                } else {
                        filename "pxelinux.0";
                }
                next-server <MAAS DHCP IF IP ADDR>;
        }

        host <호스트명>{
                hardware ethernet <PXE IF MAC ADDR>;
                fixed-address <PXE IF IP ADDR>;
        ...
}
```

그림 20 박스의 고정 IP 위한 DHCP 설정

o 새로운 박스 추가

융합형 자원 박스를 원격 자동 설치 도구를 이용하여 자동 설치/설정 하고자 하는 경우 박스를 도구에 등록해 주어야 한다. 이를 위하여 앞에 설명한 내용을 참조하여 아래 순서에 맞게 등록한다.

- 박스 추가 단계

Intel RMM 설정 -> Ubuntu MAAS 노드 설정 -> Box 설정파일 생성 (2.5장 초반부 참고) -> MAAS DHCP 설정 파일 수정 -> 박스 재부팅(Declared) -> 박스 커미셔닝(Ready)

3.3. DevStack 버전 오픈스택 Installer 개요 및 상세

3.3.1. DevStack 개요

o OpenStack이 오픈소스 기반의 클라우드 운영체제로써 날이 갈수록 많은 연구 기관, 기업에서 관심을 갖고 있다. 하지만 OpenStack 자체가 Nova, Neutron 등 다양한 프로젝트들의 집합이고 다양한 조합, 다양한 환경을 지원하기 때문에 OpenStack을 원하는 데로 동작하도록 설치하는 것 자체가 어렵기로 유명하다.

o 따라서 OpenStack을 쉽게 설치가 가능하도록 지원하는 다양한 Installation 도구들이 제안되고 있다. 이들 중 대부분은 기존의 다양한 서비스의 자동 설치를 위한 범용 서비스 Installation 도구 내에 OpenStack 설치 기능을 추가한 것이며 OpenStack의 설치를 전용으로 하는 새로운 도구들 또한 개발되어 지원되고 있다.

o Devstack은 OpenStack 전용 Installation 도구로써 근본적으로 OpenStack 상에서 서비스를 개발하여 테스트하고자 하는 개발자들이 간편하게 OpenStack 클라우드 환경을 구성할 수 있도록 돕는 것을 목표로 개발된 도구이다.

o 개발자들이 아주 편리하게 OpenStack을 설치하도록 지원하는 것이 주목적으로 설치하고자 하는 OpenStack 구성을 local.conf 라는 파일에 정의하면 이를 기반으로 Devstack이 패키지 설치, 프로젝트 설치, 구성 파일 설정 등의 전체 설치 과정을 전 자동화 한다. 즉, 사용자는 실제 설치 세부 과정까지 다룰 필요 없이 단순히 local.conf 파일만 제대로 작성하고 Devstack을 실행하면 간편하게 OpenStack 클라우드를 자동으로 설치할 수 있다. 그림 21은 실제 자동 설치 도구를 이용해 OpenStack Juno 설치 시 사용되는 local.conf 파일의 내용이다.

o OpenStack은 클라우드 기술을 대표하는 오픈소스 클라우드 운영체제이고 세계적으로 그 위상이 높으므로 Ceph, Docker, Opendaylight 등 다양한 오픈소스 프로젝트들이 이를 OpenStack과 연동시키고자 하는 움직임을 보이고 있다. OpenStack과 타 프로젝트와의 연동을 하기 위해 매뉴얼을 보며 일일이 설치하고 구성하는 것은 많은 시간과 노력을 요구한다. 따라서 일반적으로 유명한 프로젝트와의 연동에 대해서는 Devstack을 이용한 설치 및 연동 방법을 빠르게 제공하는 것이 일반적이다.

o 따라서 Devstack을 사용 시 간편하게 OpenStack 설치가 가능하며 업데이트 및 OpenStack 요소 변경에 유연하고 빠르게 대처 가능하다는 장점이 있다. 또한 타 프로젝트와의 연동을 빠르게 구성해 보고 테스트하기 위한 테스트베드 환경에서는 빠르고 간편한 Devstack이 많은 이점을 갖는다.

```
[[local|localrc]]
FLOATING_RANGE=210.114.90.0/24
Q_FLOATING_ALLOCATION_POOL=start=210.114.90.14,end=210.114.90.30
PUBLIC_IP_INTERFACE=em2
PUBLIC_NETWORK_GATEWAY=210.114.90.12

FIXED_RANGE=10.0.0.0/16
FIXED_NETWORK_SIZE=65534

MULTI_HOST=True

HOST_IP=172.30.90.12
SERVICE_HOST=172.20.90.12

MYSQL_HOST=$SERVICE_HOST
RABBIT_HOST=$SERVICE_HOST
GLANCE_HOSTPORT=$SERVICE_HOST:9292
KEYSTONE_AUTH_HOST=$SERVICE_HOST
KEYSTONE_SERVICE_HOST=$SERVICE_HOST
HEAT_API_HOST=$SERVICE_HOST
HEAT_API_CFN_HOST=$SERVICE_HOST
HEAT_API_CW_HOST=$SERVICE_HOST
CINDER_SERVICE_HOST=$SERVICE_HOST

ADMIN_PASSWORD=secrete
MYSQL_PASSWORD=secrete
RABBIT_PASSWORD=secrete
SERVICE_PASSWORD=secrete
SERVICE_TOKEN=secrete

LOGFILE=../logs/stack.sh.log
SCREEN_LOGDIR=../screen_log

disable_service n-net
disable_service n-cpu
enable_service q-svc q-agt q-dhcp q-l3 q-meta neutron
enable_service tempest
enable_service ceilometer-acentral ceilometer-anotification
ceilometer-collector
enable_service ceilometer-api ceilometer-alarm-notifier
ceilometer-alarm-evaluator
enable_service mysql

Q_PLUGIN=ml2
ENABLE_TENANT_TUNNELS=True
Q_ML2_TENANT_NETWORK_TYPE=vxlan
Q_DVR_MODE=dvr_snat

VNCSERVER_PROXYCLIENT_ADDRESS=172.20.90.12
VNCSERVER_LISTEN=0.0.0.0
NOVNCPROXY_URL=${NOVNCPROXY_URL:-"http://210.114.90.12:6080/
vnc_auto.html"}
XVPVNCPROXY_URL=${XVPVNCPROXY_URL:-"http://210.114.90.12:6081
/console"}

NOVA_BRANCH=stable/juno
GLANCE_BRANCH=stable/juno
HORIZON_BRANCH=stable/juno
NEUTRON_BRANCH=stable/juno
CINDER_BRANCH=stable/juno
KEYSTONE_BRANCH=stable/juno
CEILOMETER_BRANCH=stable/juno
```

```
[[local|localrc]]
SERVICE_HOST=172.20.90.12
MULTI_HOST=1

HOST_IP=172.30.90.35

LOGFILE=../logs/stack.sh.log
SCREEN_LOGDIR=../screen_log

MYSQL_HOST=$SERVICE_HOST
RABBIT_HOST=$SERVICE_HOST
GLANCE_HOSTPORT=$SERVICE_HOST:9292
KEYSTONE_AUTH_HOST=$SERVICE_HOST
KEYSTONE_SERVICE_HOST=$SERVICE_HOST
#CINDER_SERVICE_HOST=$SERVICE_HOST

MYSQL_PASSWORD=secrete
RABBIT_PASSWORD=secrete
SERVICE_TOKEN=secrete
SERVICE_PASSWORD=secrete
ADMIN_PASSWORD=secrete

#DATABASE_TYPE=mysql
#VOLUME_BACKING_FILE_SIZE=500G

VNCSERVER_PROXYCLIENT_ADDRESS=172.20.90.35
VNCSERVER_LISTEN=172.20.90.35
NOVNCPROXY_URL=${NOVNCPROXY_URL:-"http://210.114.90.12:6080/vnc_auto.html"}
XVPVNCPROXY_URL=${XVPVNCPROXY_URL:-"http://210.114.90.12:6081/console"}

disable_all_services
enable_service n-cpu q-agt rabbit n-novnc ceilometer-acompute q-l3
#enable_service c-vol

Q_PLUGIN=ml2
ENABLE_TENANT_TUNNELS=True
Q_ML2_TENANT_NETWORK_TYPE=vxlan
Q_HOST=$SERVICE_HOST
Q_DVR_MODE=dvr

NOVA_BRANCH=stable/juno
NEUTRON_BRANCH=stable/juno
#CINDER_BRANCH=stable/juno
```

그림 21 Devstack local.conf file (좌: Controller Node, 우: Compute Node)

3.3.2. DevStack 버전 오픈스택 Installer 상세

o DevStack 버전의 오픈스택 Installer는 현재 기본적으로 SmartX Provisioning Framework에 통합되어 있는 상태이다. 이를 Playground에 적용하기 위해서는 인프라 내 Box의 설정파일을 conf 디렉토리 내에 Box 설정 파일을 작성한 후 DevStack 기반 Installer를 적절히 활용할 수 있도록 Playground Template을 제대로 작성하면 된다.

o 다른 소프트웨어 Installer와 달리 openstack_install 디렉토리 내 바로 Installer가 위치한 것이 아니라 openstack_install/devstack 디렉토리에 위치한 이유는 오픈스택이라는 동일한 소프트웨어를 대상으로 DevStack 버전의 Installer, Full 버전 Installer의 두 Installer를 사용하기 때문이다.

o DevStack 버전 오픈스택 Installer을 활용하기 위해서는 Playground Template을 그림 22과 같이 작성해 주어야 한다. 이 때 좌측은 Controller Node를 명시한 것이고 오른쪽 양식은 Compute Node의 Template 양식을 기재한 것이다.

<Box명>{
INSTALL_SOFTWARE: openstack <추가 SW들>
OPENSTACK_MODE: Controller
OPENSTACK_INSTALLER: Devstack
}
	<Box명>{
	INSTALL_SOFTWARE: openstack <추가 SW들>
	OPENSTACK_MODE: Compute
	OPENSTACK_CONTROLLER: <Controller Node Box명>
	OPENSTACK_INSTALLER: Devstack
	}

그림 22 DevStack 버전 오픈스택 Installer를 위한 Playground Template 양식

o 위와 같이 Playground Template을 작성한 후 Framework 소프트웨어 실행하면 자동으로 prepare_devstack.sh 파일을 실행하게 된다. 이 파일은 실행 시에 대상 Box가 Controller Node인지 Compute Node인지를 전달 받아 make_local_conf.sh 파일을 호출하며 Controller/Compute 정보를 파라메터로 전달한다.

o 그러면 make_local.conf.sh 파일은 전달받은 Box의 역할을 기반으로 그림 22와 같이 Controller/Compute Node용 local.conf 파일을 Installation Coordinator 디렉토리에 복사해놓고 이어서 local.conf 파일 내 IP 주소 등의 Box별 설정 값을 Box의 설정파일을 참고하고 수정해 놓는다.

o 그리고 이어서 Box에 복사되어 설치과정을 진행할 Installer 파일들을 Installation Coordinator 디렉토리에 복사해 놓아 향후 Installation Coordinator가 네 번째 단계에서 Installer와 local.conf 파일을 Box에 복사한 후 실행함으로써 DevStack 도구를 활용한 오픈스택 자동설치가 자동으로 완료된다.

3.4. Full 버전 오픈스택 Installer 개요 및 상세

3.4.1. 개요

o 개발된 DevStack 버전의 오픈스택 Installer를 활용하여 SmartX Playground를 대상으로 멀티사이트 오픈스택 클라우드를 자동으로 설치/설정하여 활용하였다. 하

지만 DevStack 이라는 도구 자체가 오픈스택 Developer들이 오픈스택 테스트를 위해 사용하는 도구이기 때문에 신속하고 쉽게 오픈스택 클라우드를 설치/설정할 수 있다는 장점이 있지만 반대로 안정성이 요구되는 운용 측면에서는 많은 한계점을 드러내었고 이로 인하여 많은 어려움을 겪었다.

o 이러한 DevStack을 이용해서 설치함으로 인해 발생되는 문제점들을 해소하고자, Ubuntu의 APT 등을 활용하여 멀티사이트 SmartX Playground를 대상으로 보다 안정적으로 오픈스택을 설치/설정할 수 있는 방법에 대해 연구하였다. 이러한 연구의 결과로 GIST, KOREN NOC, 제주대, 고려대, 포항공대 사이트에 위치한 SmartX Box들을 대상으로 오픈스택을 자동 설치/설정할 수 있는 스크립트 기반의 도구를 개발하였다.

o 본 도구를 활용하여 설치한 오픈스택 클라우드 환경이 기존 DevStack로 설치한 환경에 비하여 많은 부분에서 안정화되었기 때문에 현재 GIST, KOREN NOC, 제주대, 고려대에 위치한 5개의 SmartX Box을 엮어 오픈스택 클라우드 환경을 구성하여 다수의 연구자들을 대상으로 멀티사이트 오픈스택 클라우드 가상놀이터 환경을 제공하고 있는 상황이다.

o 해당 Installer는 SmartX Playground에서 요구하는 모든 환경을 매우 안정적인 형태로 설치/설정해주기 때문에 DevStack 기반 Installer와 구분하기 위하여 오픈스택 Full 버전 Installer라고 명명하였다.

3.4.2. Full 버전 오픈스택 Installer 상세

o Full 버전 오픈스택 Installer 또한 현재 SmartX Provisioning Framework 내에 이미 통합되어 있는 상태이므로 이 Installer를 활용하여 Provisioning을 하고자 할 경우 Playground Template을 정해진 규격에 맞춰 작성하기만 하면 된다.

<Box명>{
INSTALL_SOFTWARE: openstack <추가 SW들>
OPENSTACK_MODE: Controller
OPENSTACK_INSTALLER: Full
}
	<Box명>{
	INSTALL_SOFTWARE: openstack <추가 SW들>
	OPENSTACK_MODE: Compute
	OPENSTACK_CONTROLLER: <Controller Node Box명>
	OPENSTACK_INSTALLER: Full
	}

그림 23 Full 버전 오픈스택 Installer를 위한 Playground Template 양식

o Full 버전 Installer의 Template 양식은 그림 23와 같다. 이를 보게 되면 DevStack 버전 Installer와 크게 다른 것이 없으며 오직 다른 부분은 OPENSTACK_INSTALLER 설정 값을 DevStack이 아닌 Full로 설정한다는 것이다. 단지 이 설정 값을 조정함으로써 오픈스택 설치를 위해 사용하는 Installer를

DevStack 버전 또는 Full 버전 중 쉽게 선택할 수 있다.

o 구현한 Full 버전 Installer의 상세 동작원리를 그림 24로 정리해 보았다. 그림에서 초록색 라인이 Controller Node 설치/설정 시 자동으로 수행되는 과정이고 하늘색 라인이 Compute Node 설치/설정을 위한 과정이다.

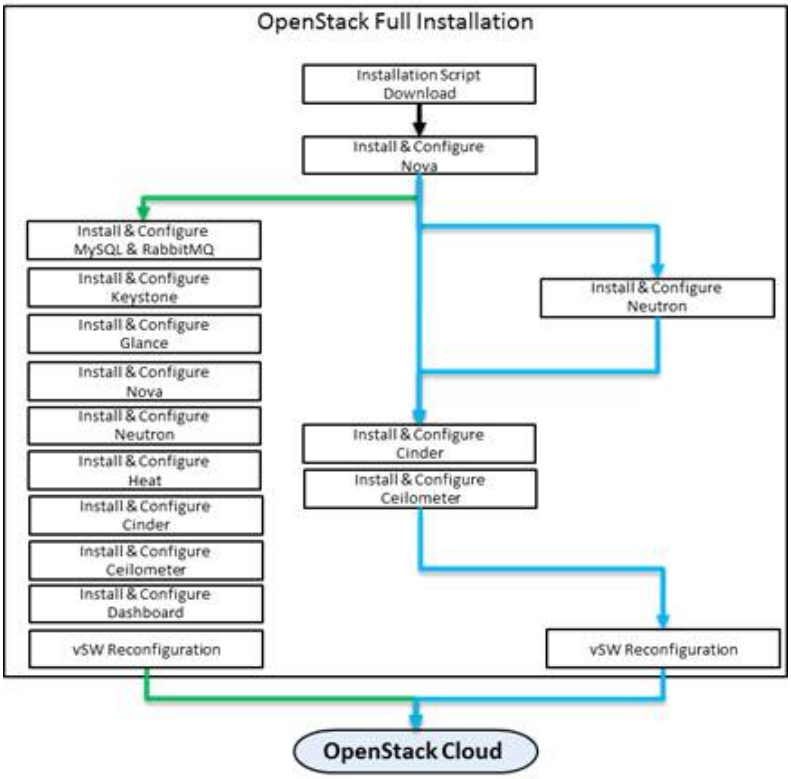

그림 24 Full 버전 오픈스택 Installer 설치/설정과정

o 이 때 Compute Node 설치 과정에서 Nova와 Cinder 사이에 2가지 갈래는 OpenStack Juno 버전에서 새롭게 도입된 Distributed Virtual Router(DVR) 기능의 Enable/Disable의 여부에 따라 갈리게 된다. DVR이 Enable 될 경우 Compute Node에 오픈스택 Neutron L3 Agent가 설치되므로 이를 위한 추가 설치/설정작업이 요구되고 DVR이 disable 될 경우 오픈스택 Neutron의 설치/설정이 필요하지 않다. 기본적으로 SmartX Playground에서는 DVR이 default로 Enable 되어 있다.

4. SmartX Provisioning Framework SW 검증

4.1. Ubuntu + OpenStack 설치 시간 측정

o 상기와 같이 구성된 원격 자동화 설치 도구의 실제 동작하는 단계를 제시함으로써 원격 자동화 설치 기능을 검증한다.

o 검증을 위한 시나리오는 아래 그림과 같다. 자동화 도구를 이용하여 광주 사이트의 SmartX Box 상에 Ubuntu와 OpenStack 클라우드를 자동 설치한다.

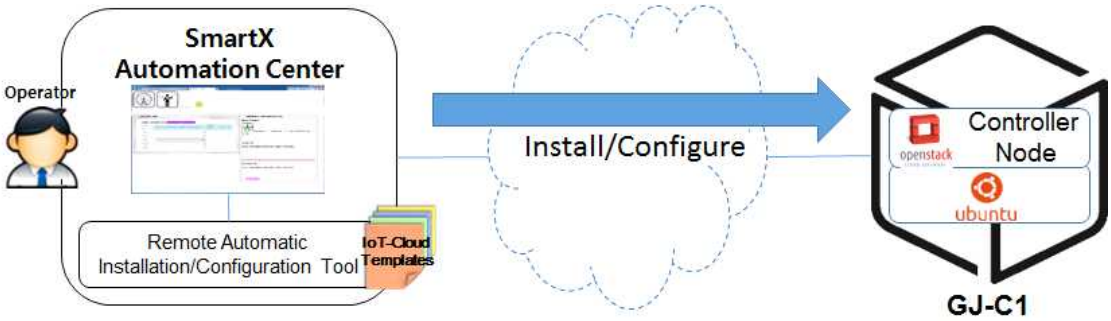

그림 25 원격 자동 설치 검증 시나리오

o OF@KOREN 테스트베드에서는 SmartX Automation Center를 통해 구현된 도구들을 활용하여 중앙 집중적으로 프로비저닝, 오케스트레이션 등을 Web UI 기반으로 간편하게 제어가능하다.

원격 자동 설치 도구 또한 SmartX Automation Center에서 활용하는 도구 중 프로비저닝 도구에 해당하며, 실제로는 직접 템플릿을 작성하고 실행하여 원격 설치가 가능하나 SmartX Automation Center를 통해 이 과정을 간편화 할 수 있다.

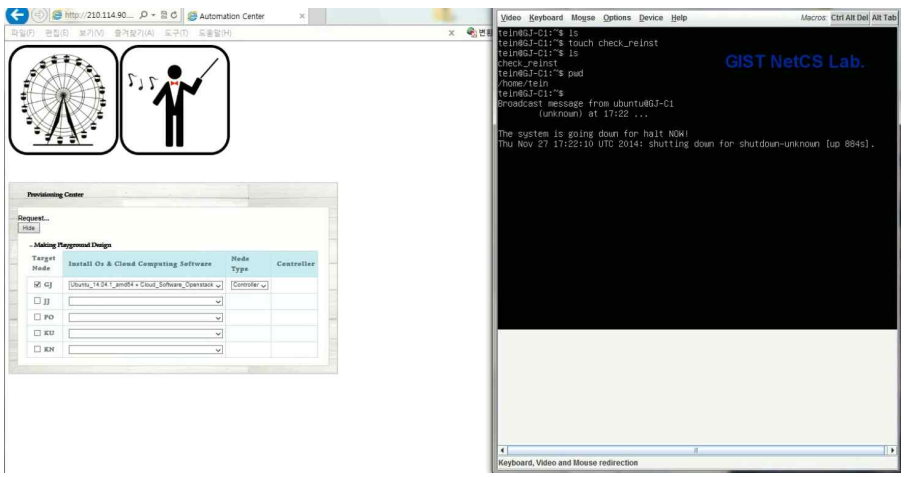

그림 26 SmartX Automation Center UI

o 그림의 좌측이 SmartX Automation Center의 Web UI 이며, 우측이 광주 SmartX Box의 RMM 화면을 나타낸다. 좌측 하단과 같이 설치 대상 박스, 설치할 소프트웨어, 해당 노드의 역할(OpenStack Controller/Compute) 등을 지정하여 Start 버튼을 누르게 되면 자동으로 템플릿을 생성하고 설치를 시작한다.

o 설치 시작 시 MAAS가 IPMI의 전원 관리 기능을 이용하여 원격의 박스의 전원을 재부팅 시킨다.

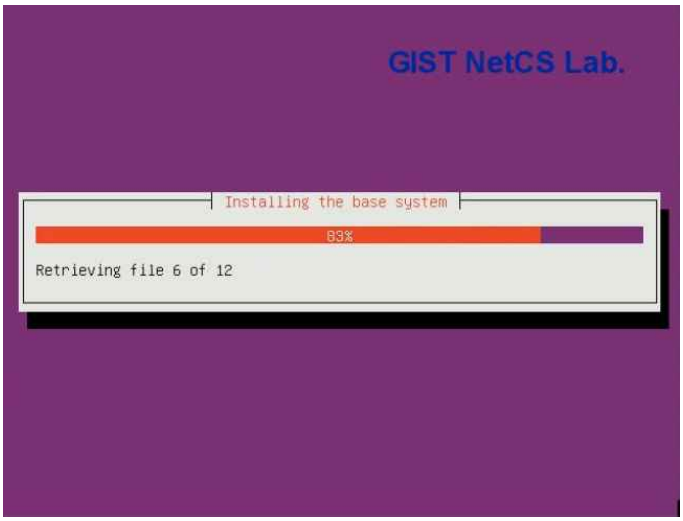

그림 27 Ubuntu 설치 화면

o 그림은 재부팅 후 자동으로 Ubuntu 설치가 진행되는 모습을 나타낸다.

그림 28 기본 설정 및 OpenStack 설치 시 화면

o Ubuntu 설치가 완료되면 자동으로 박스가 다시 재부팅 되며 그림과 같은 화면과 함께 Ubuntu 부팅 과정이 진행된다. 실제 화면은 위와 같이 나타나나 Ubuntu

부팅 과정 이후 계정 설정, 네트워크 인터페이스 설정, 필수 패키지 설정, Apt 설정 등이 모두 자동으로 이루어지며 곧바로 이어서 Devstack 설치, OpenStack 설치까지 전체 과정이 자동으로 진행된다.

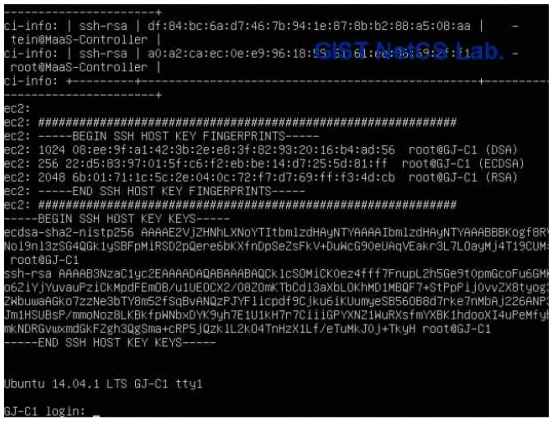

그림 29 설치 완료 시 화면

o 템플릿에 지정된 모든 소프트웨어의 설치/설정이 마무리되면 Ubuntu 부팅 화면이 전환되어 위의 로그인 화면이 나타난다.

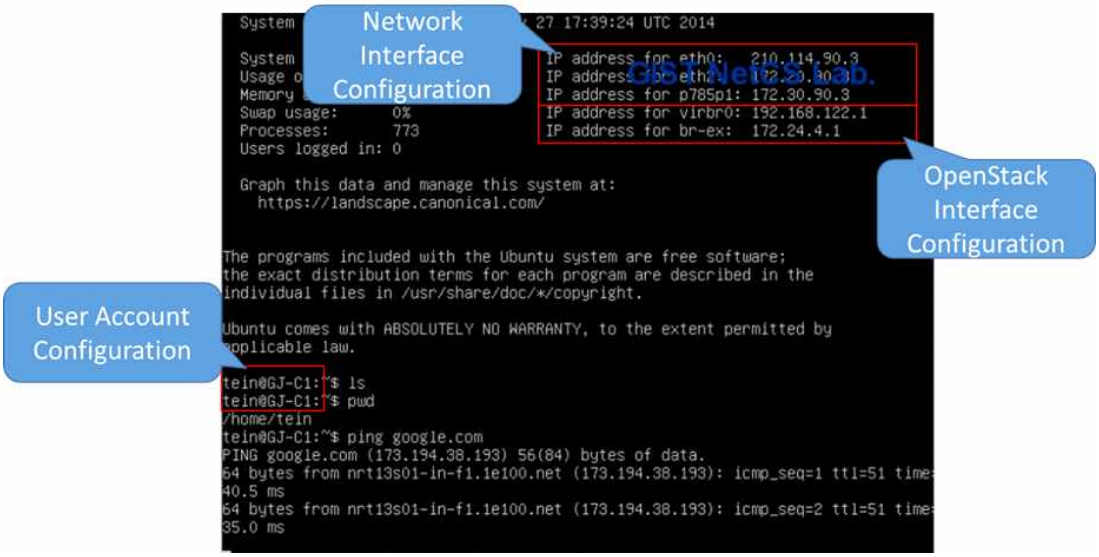

그림 30 설치 결과 검증

o 그림을 통해 자동 설치/설정 도구의 실행 결과로 네트워크 인터페이스 설정, 사용자 계정, OpenStack 설정까지 자동으로 진행되었음을 확인할 수 있다.

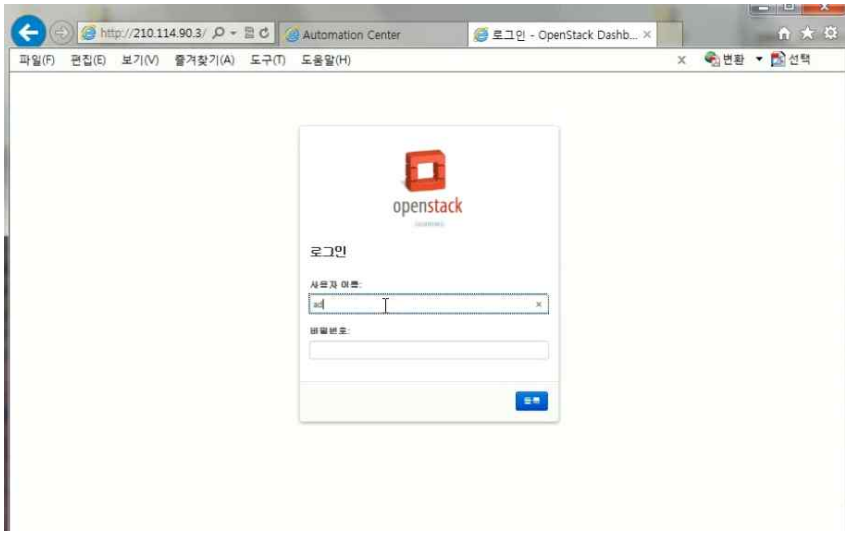

그림 31 자동 설치된 OpenStack의 Horizon Dashboard

o 설치 완료 이후 “http://<컨트롤러 노드 관리 IP>”를 통해 OpenStack Horizon Dashboard에 접근 가능하다.

설치 직후 관리자 계정의 ID/PW는 admin/secrete 로 자동 구성되며 설치 이후 변경하여 사용할 것을 추천한다.

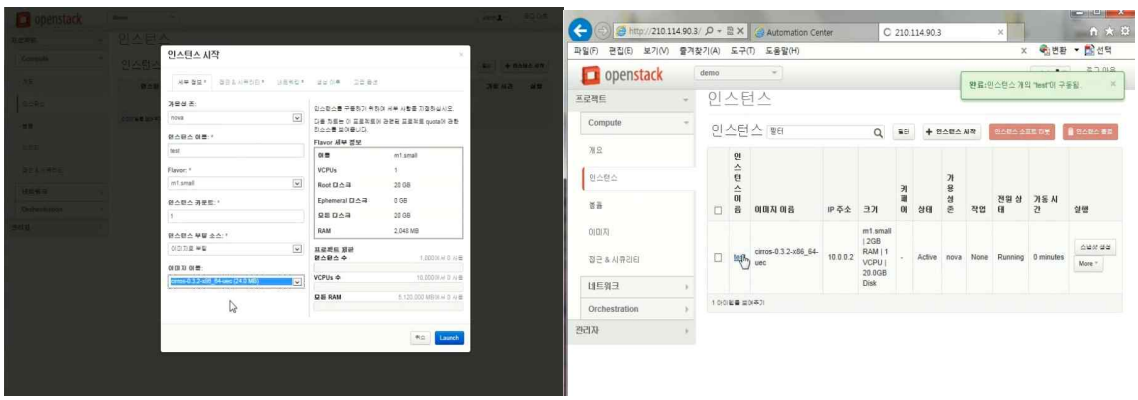

그림 32 자동 설치된 OpenStack 내 VM 생성 검증

o 그림을 통해 설치된 OpenStack 클라우드 상에 VM을 정상적으로 생성하여 활용 가능함을 볼 수 있다.

References

[1] DevOps, http://en.wikipedia.org/wiki/DevOps
[2] IPMI, http://en.wikipedia.org/wiki/Intelligent_Platform_Management_Interface
[3] J. Kim et. al., "OF@TEIN: An OpenFlow-enabled SDN testbed over international SmartX Rack sites," in *Proc. APAN –Networking Research Workshop*, Aug, 2013.
[4] Devstack, http://docs.openstack.org/developer/devstack/
[5] Ubuntu MAAS, http://maas.ubuntu.com
[6] OpenStack, http://openstack.org
[7] J. Shin and J.Kim, "Functionality verification of automated remote installation of converged resource boxes for distributed cloud," *in Proc. 27th Korean Signal Processing Conference (KSPC)*, Seoul, Korea, Sep. 2014
[8] PXE, http://en.wikipedia.org/wiki/Preboot_Execution_Environment

오픈 SmartX 플랫폼에 통합된 SmartX Box의 원격 자동설치 및 제어

- 이 문서의 기술적인 내용은 OF@KOREN 프로젝트의 진행과 함께 별도의 예고 없이 변경될 수 있습니다.
- 최신의 정보로 갱신된 매뉴얼에 대한 정보와 매뉴얼에 대한 문의 사항은 아래의 정보를 참조하시길 바랍니다.
  - Homepage: https://nm.gist.ac.kr
  - E-mail: ops-koren@smartx.kr

작성기관: 광주과학기술원
작성년월: 2015/12