시스템 설정 및 관리를 위한 자동화 도구 분석
-Puppet, Chef, Heat-

부　서: KISTI 국가슈퍼컴퓨팅연구소
작성자: 조혜영, 신승찬, 염재근, 함재균

한국과학기술정보연구원

Korea Institute of Science and Technology Information

목 차

제 1장 서론

기존에는 슈퍼컴퓨터(supercomputer)를 컴퓨터의 개발목적이었던 군사, 경제 관련의 과학기술연산 분야만을 목적으로 이용되었다. 하지만 2000년대에 와서는 그 활용분야가 대폭 늘어났고 보다 많은 기업, 집단들이 슈퍼컴퓨팅 기술을 요구하게 되었다. 늘어난 수요에 보다 더 쉽고 간편한 서버관리 도구가 요구되었고, 그로 인해 다양한 자동화된 소프트웨어 형상관리(software configuration management) 도구가 개발되었다.

'형상관리' 란 형상(configuration) 항목을 식별하여 그 기능적 물리적 특성을 문서화하고 그러한 특성에 대한 변경을 제어, 변경 처리 상태를 기록 및 보고, 더불어서 명시된 요구사항에 부합하는지 확인하는 기술적이고 관리적인 감독, 감시 활동을 뜻한다 [0]. 소프트웨어 개발은 어느 단계에서나 변경이 일어날 수 있으므로 형상관리는 매우 중요하다. 설령 개발이 끝난 후에도 지속적인 업데이트가 있을 경우 역시 형상관리가 필요하다.

유저당 하나의 노드에 설치된 소프트웨어는 형상관리가 비교적 쉽다. 하지만 슈퍼컴퓨팅 서비스 같은 유저에 비해 노드 수가 압도적으로 많을 경우 배포 단계부터 만만치 않게 된다. 적게는 100개의 노드부터 많게는 몇 천 개의 노드를 일일이 설정하기에는 시간과 비용이 너무나 많이 소비된다. 혹여나 에러가 날 경우 이전과 같은 지루한 작업을 다시 반복해야 된다. 이러한 반복적인 작업을 간소화 시켜주는 것이 바로 형상관리 자동화 도구(automation tool) 이다.

현재 한국과학기술정보연구원의 슈퍼컴퓨팅연구소는 오픈스택(Openstack) 이라는 IaaS 형태의 클라우드 컴퓨팅 오픈소스 프로젝트를 이용하고있다. Openstack 으로 형성한 node에 클라이언트가 요구하는 환경 구축 및 application 설치 작업을 간소화 해줄 형상관리 도구로 Puppet 또는 Chef를 염두에 두고 있다. 또한 오픈스택이 제공하는 Orchestration 도구인 Heat 과 더불어 보다 간편하고 저렴한 도구를 선택하려고 한다.

본 보고서는 다음과 같이 구성된다. 2, 3장에서는 각각 Puppet과 Chef 에관한 간단한 개요 및 설치, 실행방법을, 4장에서는 오픈스택 고유의 orchestration 도구인 Heat 에 관한 설명 및 설치, 그리고 puppet 과

chef와 연계하는 법을 기술하였고, 5장은 이러한 도구들을 비교 분석하여 최종적으로 6장에 결론을 기술하였다.

제 2장 Puppet

2.1 Puppet 개요

Puppet 이란 Luke Kanies가 2005년에 설립한 Puppet Labs에서 개발한 형상관리 도구로 인프라 자동화의 심플함과 생산성을 추구하기 위해 개발되었다. Ruby로 개발되었고, 자체 시스템 구성 언어를 써서 개발자가 굳이 어떻게 프로그램이 실행되는지 구체적으로 알 필요 없이 각 노드들에 필요한 서비스, 패키지를 관리 할 수 있게끔 해준다. 이전에 존재한 자동화 도구인 CFEngine 2 를 대체할 수 있도록 개발되었으며 유저가 '어떻게' 이 툴을 사용해서 시스템을 관리할 것인가 보단 '무엇을' 하고 싶은지에 초점 되어있다. 현재 안정화된 3.6.2 버전까지 나온 상태다. 오픈소스 버전과 유료인 엔터프라이즈 버전 형태로 제공된다.

2.1.1 Open source vs. Enterprise

FEATURES	Open Source Puppet	Puppet Enterprise
Graphical User Interface		✔
Event Inspector - Visualize Infrastructure Changes		✔
Supported Modules		✔
Provisioning – Amazon EC2	✔	✔
Provisioning – Google Compute Engine	✔	✔
Provisioning – VMware VMs		✔
Configuration management – Discovery		✔
Configuration management – User accounts		✔
Configuration management – Operating systems & applications	✔	✔
2,000+ pre-built configurations on Puppet Forge	✔	✔
Orchestration – Task automation		✔
Role-Based Access Control – Now with external authentication support		✔
Unified cross-platform installer of all components		✔
Support – Option for 24 x 7 x 365		✔
Support – Defined SLA		✔
Certified by Puppet Labs engineers		✔
Pre-packaged dependencies in one directory		✔
Smooth upgrade and maintenance path		✔

Table 1. Open Source Puppet 과 Puppet Enterprise 가 제공하는 기능 비교

무료 (open source) 버전과 유료 (enterprise) 버전의 서비스 차이는 위와 같다. 프로그램 자체의 성능은 무료나 유료나 비슷하나 프로그램의 부가적인 기능 이외에도 유료버전을 이용함으로써 Puppet labs 의 지원을 받거나 공인된 module 을 사용할 수 있는 혜택이 주어진다. 지원(support) 서비스를 받지는 못하지만 유료버전을 무료로 10노드까지 사용할 수 있다.

유료버전은 두 가지로 나뉘는데 기본적인 지원 (standard support) 서비스와 프리미엄 지원 (premium support) 서비스 플랜이 존재한다.

Enterprise Support Tiers

	Standard	Premium
Escalated Bug Fixes	✔	✔
Support Hours	6am – 6pm PST, Monday – Friday	24 x 7 x 365 for Priority 1
Email Support	✔	✔
Phone Support	✖	✔
Access to Private Support Portal & Forums	✔	✔
Feature Priorities	✖	✔
Access to All Updates & Upgrades of Puppet Enterprise	✔	✔
Maximum number of technical contacts	Four (4)	Unlimited
Response Times	Priority 1: 1 business hour Priority 2: 4 business hours Priority 3: 12 business hours	Priority 1: 1 calendar hour (contact via phone) Priority 2: 4 business hours Priority 3: 12 business hours
Number of Cases/month	5	Unlimited
Free Public Training	✖	Four (4) Engineers

Table 2. Standard Support 와 Premium Support 비교

위의 그림에서 볼 수 있듯이 프리미엄 서비스는 보다 더 적극적인 지원 서비스를 제공한다. 유저는 프리미엄 서비스를 이용함으로써 전화 상담이 가능해지고 365일 24시간 언제든지 상담 받을 수 있게 된다. 그리고 유저의 특정 기능 요구(request)에 대해 그 기능이 다음 버전에 추가될지를 우선적으로 고려 받을 수 있게 된다. 또한 최대한 받을 수 있는 기술적 지원 서비스는 무제한이고 한달 동안 받을 수 있는 지원 역시 무제한이다. 그리고 4명까지의 기술자들을 무료로 연수 받게 할 수 있다.

Puppet Enterprise Pricing

Cumulative # of Nodes	Per Node Standard Support	Per Node Premium Support*
1-10	Download FREE	N/A
11-99	$112	Contact Sales
100-249	$105	$199
250-499	$99	$135
500-999	$95	$119
1000-2499	$93	$112
2500+	Contact Sales	Contact Sales

Table 3. Puppet Enterprise 가격

유료버전의 가격은 사용하는 노드에 따라 달라진다. 또한 일반적인 지원(Standard Support)서비스 보다 프리미엄 지원서비스가 가격이, 노드 100개 기준으로, 대략 두 배 가량 비싸다. 하지만 지원받는 노드 수가 많을수록 일반지원과 프리미엄 지원의 가격차이가 급격히 줄어든다.

2.1.2 커뮤니티

무료버전인 open source Puppet 을 쓰는 유저는 Enterprise버전처럼 Puppetlabs의 공식적인 지원을 받을 수 없다. 하지만 그러한 유저들을 위해 다양한 커뮤니티가 존재한다. 아래는 공식사이트에 기재된 커뮤니티들이다:

- Mailing list
- Puppet open source 유저 그룹
- Puppet 개발자 그룹
- Q&A 사이트
- #puppet 혹은 #puppet-dev irc 채널
- Puppet labs 티켓

Puppet open source 유저 그룹과 Puppet 개발자 그룹은 둘다 구글(Google) 그룹스로 gmail 아이디가 있는 유저는 그룹에 가입한 후 자유롭게 기존 토

픽을 읽거나 새 토픽을 제시 할 수 있다. Q&A 사이트는 puppet의 ask 사이트로 깔끔한 UI를 가지고 있어 한눈에 질문과 답변상황을 알기 편하다. 직접 질문 글을 올리려면 로그인 이 필요하지만 AskPuppetLabs 계정을 만들지 않아도 트위터나 구글, 야후, OpenID 계정이 있으면 로그인이 가능하다.

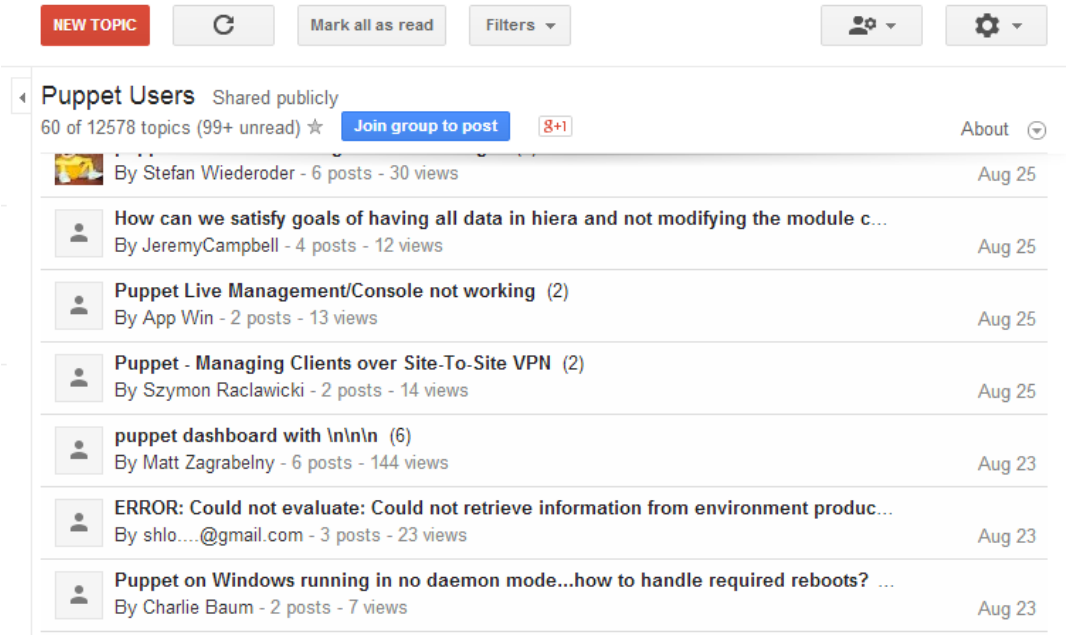


Figure 1. 구글 Puppet Users Group

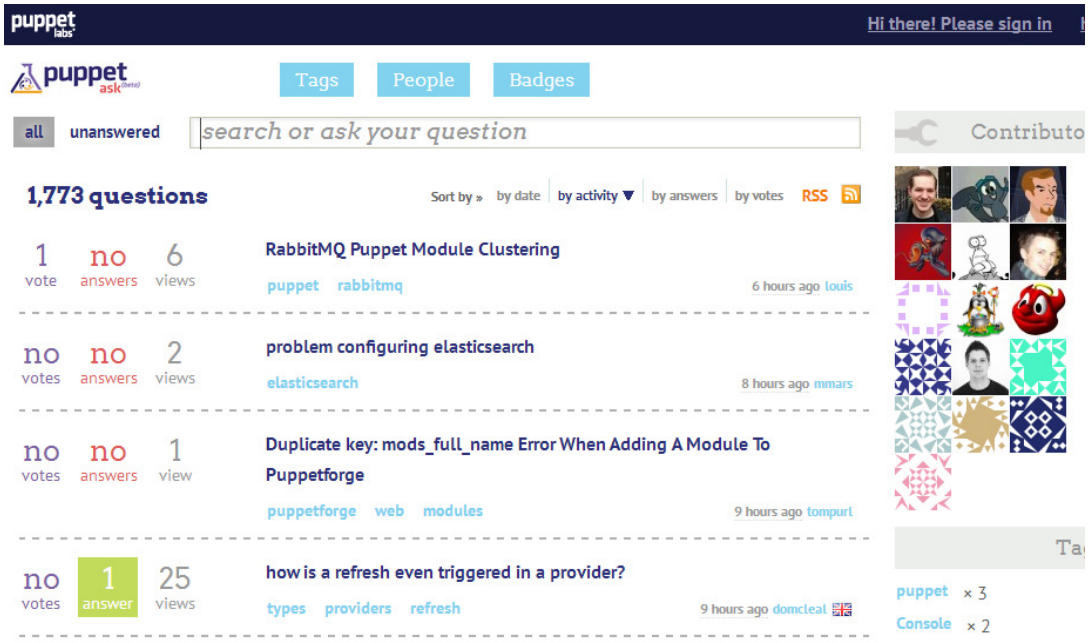


Figure 2. Ask Puppet Lab 홈페이지

IRC 채널의 경우 온라인 채팅 사이트로 다양한 Puppet 유저들이 실시간으로 의견을 주고받고 질문하거나 답변할 수 있는 장점이 존재한다. 하지만 실시간이기에 제대로 읽히지 않고 넘어가 버릴 수 있기에 이럴 땐 ask 사이트나 mailing list 를 이용하는 것이 좋다.

2.2 Puppet 구조

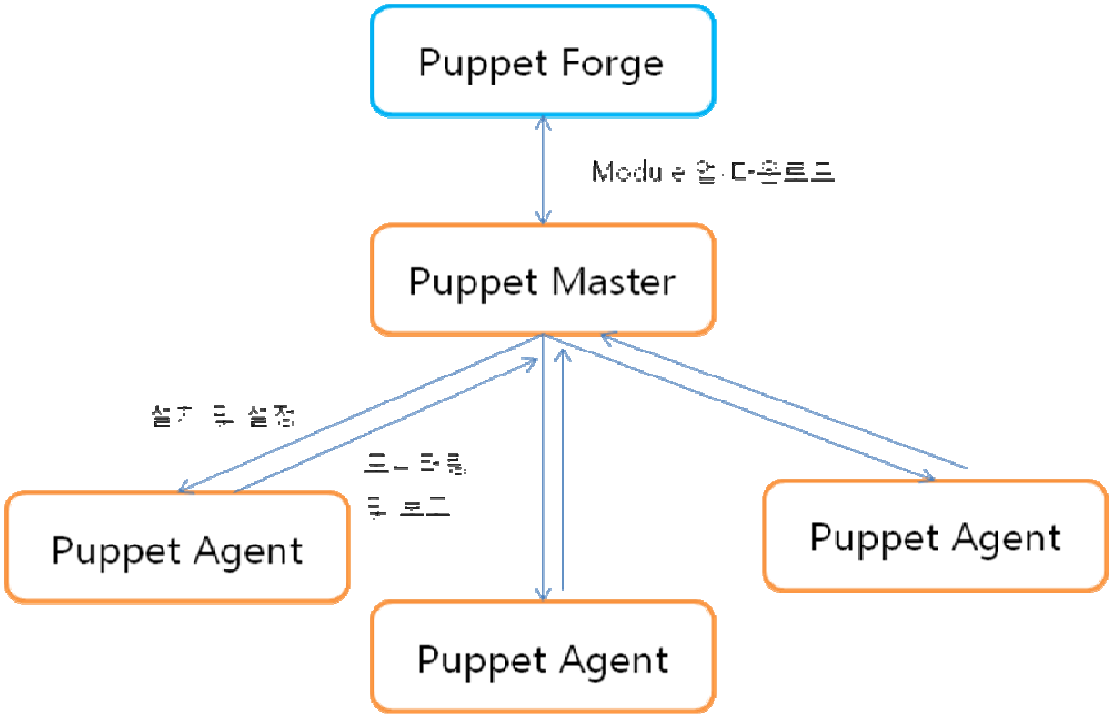


Figure 3 Puppet 이 형상관리하는 구조

Puppet 은 server-client 구조를 가지고 있고 Sever는 Master, client는 Agent 라고 불린다. Master 에서 module을 작성하고 배포하면 agent는 설치 후 master에게 보고한다.

2.3 Puppet 설치

2.3.1 설치 개요

Open source Puppet 의 경우 yum 을 통해 간편하게 패키지 다운로드가 가능하며, master 와 agent 노드에 설치되는 패키지는 각각 puppetmaster 와 puppet 이다. 설치 과정의 흐름은 간략하게 아래와 같다:

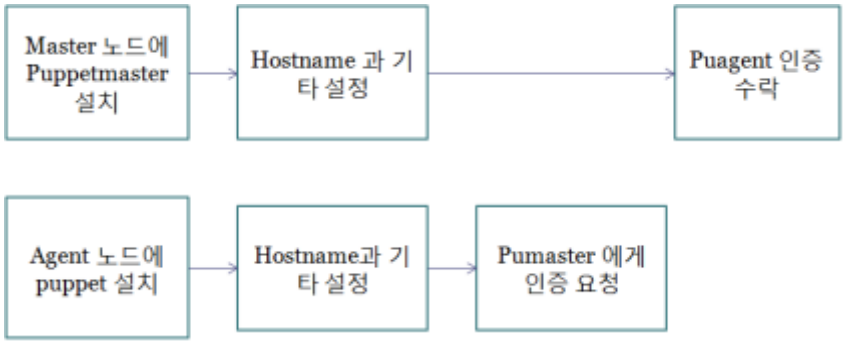


Figure 4. Puppet 설치 과정

위의 흐름 대로 1개의 master와 11개의 agent 에 서버를 설치하였으며 설치 환경은 아래와 같다.

	Master 노드	Agent 노드
설치된 OS	Centos 6.5	Centos 6.5
노드 이름 (hostname)	pumaster	puagent1 ~ puagent11
할당된 ip	172.16.10.3	172.16.10.4~5, 172.16.10.8~16

Table 4. Puppet Master 과 Agent 설치 환경

2.3.2 설치 방법

yum 이 아니더라도 tar.gz 압축 파일을 usb로 옮겨와 설치가 가능하지만 편의상 yum 으로 패키지를 다운받아 설치하였다.

☐ yum install

각 노드에따라 알맞은 패키지를 설치한다

Master 노드

```
# yum install -y puppet-server
```

Agent 노드

```
# yum install -y puppet
```

☐ hostname 설정

각각의 노드의 이름을 설정해준다

```
# hostname nodename
```

nodename 대신 노드에 알맞은 이름을 넣어주면 된다. 필자의 경우 Master 노드는 pumaster.example.com, Agent 노드들은 puagent1 ~ puagent11.example.com으로 설정해주었다.

☐ /etc/hosts 설정

Agent들이 Master에게 인증을 받기 위에서는 /etc/hosts에 서로에 대한 도메인 정보가 있어야 한다.

```
Master 의 경우
172.16.10.3 pumaster.example.com
172.16.10.4 puagent1.example.com
172.16.10.5 puagent2.example.com
…
172.16.10.16 puagent11.example.com

Agent 의 경우
172.16.10.3. pumaster.example.com
```

Master의 경우 Agent 노드들의 정보를 다 입력해줘야 하고, Agent는 Master 노드 정보만 있으면 된다.

☐ /etc/sysconfig/network 에서 hostname 설정

각 노드마다 앞에서 설정해준 노드 이름을 입력해준다.

```
HOSTNAME = nodename
```

☐ Agent 노드의 /etc/puppet/puppet.conf 설정

아래와 같이 추가해주면 된다:

```
[agent]
server = Master_nodename
report = true
pluginsync = true
certname= Agent_nodename
```

Master_nodename 에는 Master 노드의 hostname을, *Agent_nodename* 에는 해당 Agent의 hostname을 입력하면 된다. 필자의 경우 각각 pumaster.example.com, puagent1.example.com을 입력했다.

□ 8140 포트

Master 노드가 Agent 노드의 인증신청을 받을 수 있게 8140 TCP 포트를 열어줘야 한다. Openstack instance 들을 노드로 쓰고있는 경우 Dashboard 의 접근&보안 에서 규칙편집으로 추가해 주면 된다.

□ 인증 요청

우선 Master 노드에서 아래 커맨드로 퍼펫 서버를 가동시킨다:

```
# puppet master --no-daemonize --verbose
```

그다음 Agent 노드에서 아래 커맨드로 퍼펫을 가동시킨다:

```
# puppet agent --server master_nodename --no-daemonize --
verbose
```

이때 --no-daemonize 는 puppet 을 데몬이 아닌 foreground 로 실행하는 옵션이고, --verbose는 verbose 모드로 실행하라는 뜻이다.

위를 마쳤으면 Master 노드에서 Agent들의 요청이 들어오는 것을 볼 수 있을것이다.

□ 인증 발급

Master 노드에서 아래와 같은 커맨드를 입력한다:

```
# puppet cert list
# puppet cert --all and --list
```

위의 커맨드는 아직 인증완료가 되지 않는 Agent만을, 아래는 인증신청이 들어온 모든 Agent들을 표시해준다.

```
  "puagent.example.com #jungeun3.novalocal" (SHA256) 00:E9:CA:CC:C7:A9:82:91:0
D:4F:93:74:E5:C5:6D:CA:75:E2:FE:1D:A2:D2:D8:44
+ "puagent1.example.com"                     (SHA256) 34:FA:63:C0:AB:F9:D8:BC:B
B:42:D6:7C:5C:12:B8:5D:A5:8E:F6:70:10:E8:93:13
+ "puagent10.example.com"                    (SHA256) 70:C1:05:DE:A9:34:E1:61:1
0:1F:16:0C:84:F8:B6:0D:0B:F4:CA:20:98:4D:77:6C
+ "puagent11.example.com"                    (SHA256) 1B:F2:A6:DD:03:DD:05:64:A
C:0D:D5:EA:AF:DB:C2:86:FB:74:AF:4E:5D:4B:B0:08
+ "puagent2.example.com"                     (SHA256) 51:5E:C6:D5:22:17:54:98:D
1:91:5B:42:CC:F9:FD:81:36:E9:D6:FC:32:DF:FF:DC
+ "puagent3.example.com"                     (SHA256) 4F:AD:8F:81:5C:93:EE:0C:8
7:C4:75:98:CD:65:D9:97:CB:7F:AC:AD:0B:0B:3F:2B
+ "puagent4.example.com"                     (SHA256) 05:3F:CC:49:55:D3:8F:F7:C
0:4A:B1:DB:E7:EF:33:95:B8:84:F4:BC:D1:51:85:A1
```

Figure 5. list of certification requests

“+” 표기된 부분은 이미 발급이 완료된 Agent들을 나타낸다.

Master에서

```
# puppet cert sign Agent_nodename
```

로 해당 Agent의 인증을 허락하고, Master, Agent 노드 각각에

```
Master
# puppet resource service puppetmaster ensure=running enable=true

Agent
# puppet resource service puppet ensure=running enable=true
```

를 입력해줌으로서 puppet master/agent 설치,설정, 및 실행이 완료된다.

2.3.3 오류 및 해결법

- **Error:Could not run: Could not create PID file…**
  Master/Agent 가 ensure=running, enable=true 된 상태에서 퍼펫을 구동시킬 때 뜨는 에러로 # ps -ax | grep puppet 으로 이미 실행된 것을 kill 해주면 된다.

- **Agent 의 인증요구를 Master가 인식하지 못함.**
  Agent가 인증요청을 했는데도 # puppet cert list 에 뜨지 않는 경우가 있는데 이는 8140 포트가 닫혀있거나 /etc/hosts/ 안의 hostname 불일치 때문일 수 있다.

2.4 Module

모듈은 Puppet 을 통해 프로그램 설치 및 설정에 필요한 코드와 데이터를 보관하고 있는 집합체이다.
아래는 간단한 Module 레이아웃에 대한 설명이다:

Module_name – 가장 밖에 있는 디렉토리로 Module 이름과 일치해야한다.

- manifests – Module 이 어떻게 설치되어있는지에 관한 모든 manifests를 포함하고 있다.
- files – 노드가 내려받을 수 있는 정적(static) 지원 파일을 포함하고 있다.
- templates – Module manifests 가 사용할 수 있는 templates를 포함하고 있다.
- lib – 플러그인들을 포함하고 있다.
- facts.d – 외부 변수들을 포함하고 있다.
- tests – 어떻게 module 클래스과 정의된 타입들을 선언할 수 있는지에 대한 예시들을 포함하고 있다.
- spec – lib 디렉토리 안에 있는 플러그인들을 위한 spec tests 를 포함하고 있다.

위는 기본 레이아웃이고 실제 module 에는 위의 항목들이 다 들어 있지 않을 수도 있다. 아래는 Puppet Forge 에서 받은 ntp module 의 구성 레이아웃이다.

```
├── manifests
│   ├── init.pp
│   ├── ntpdate.pp
│   └── params.pp
├── metadata.json
├── Rakefile
├── README.md
├── spec
│   ├── spec_helper.rb
│   └── spec.opts
├── templates
│   ├── ntp.conf.erb
│   ├── ntpdate.defaults.erb
│   ├── ntp.defaults.debian.erb
│   ├── ntp.defaults.redhat.erb
│   └── step-tickers.erb
└── tests
    ├── init.pp
    └── ntpdate.pp
```

Figure 6. ntp module 의 tree 레이아웃

2.4.1 Manifests

'Manifests'는 설정 규칙을 포함하고 있는 파일로 '.pp' 확장명을 쓴다. Manifests 안에는 어떤 자원이 어떠한 상태로 존재하길 원하는지 명시되어 있다. Manifests 들은 하나의 클래스나 정의된 타입을 가지고 있어야 하는데 init.pp 의 경우 반드시 클래스 이름이 module 이름과 같아야 하며 다른 manifests들은 각자의 이름과 같은 이름의 클래스를 가지고 있어야 한다.

```
class ntp(
  $server_list = [
    '0.pool.ntp.org',
    '1.pool.ntp.org',
    '2.pool.ntp.org',
    '3.pool.ntp.org',
  ],
```

Figure 7. ntp/manifests/init.pp 의 class

```
file { $config_file:
  ensure  => $ensure,
  owner   => 0,
  group   => 0,
  mode    => '0644',
  content => template('ntp/ntp.conf.erb'),
  require => Package[$package],
}
```

Figure 8. ntp/manifests/init.pp 의 자원 상태 명시(resource declaration) .

2.4.2 Files

Files 는 자동적으로 agent 노드를 위해 제공된다. puppet:///URLs 을 이용해 내려 받을 수 있으며 URLs 에 들어갈 수 있는 포맷은 아래와 같다:

Protocol	3 slashes	“Modules”/	Name of module/	Name of file
puppet:	///	modules/	my_module/	service.conf

예를 들어 puppet:///modules/my_module/service.conf 는 Master 노드의 my_module/file/service.conf 로 연결시켜준다.

2.4.3 Templates

어떠한 ERB 템플릿이라도 template function 을 통해 manifest 안에 생성이 가능하다

Protocol	3 slashes	“Modules”/	Name of module/	Name of file
puppet:	///	modules/	my_module/	service.conf

예를 들어 template(‘my_module/component.erb’) 는 template/my_module /templates/component.erb 를 생성한다.

Template 은 client 가 아닌 parser 에 의해 평가된다. 이는 곧 client 가

굳이 template 을 다운받지 않아도 된다는 뜻으로, template은 master server에만 존재하면 된다.

제 3장 Chef

3.1 Chef 개요

Chef 는 2009년 Opscode라는 회사에서 형상관리 및 시스템 통합의 자동화 목적으로 개발한 오픈소스 프레임워크다[1]. 다른 설정 자동화 툴에 비해 최근에 개발된 후발주자이지만 짧은 시간 안에 시장에 성공적으로 안착하였다. Mercado Libre, Admeld, Prezi, Facebook 과 같은 유명기업이 Chef를 이용하고 있다.

Chef는 "infrastructure as code" 와 "There's more than one way to do it" 을 모토로 삼아 디자인 되었다. 그렇기에 유저가 인프라(infrastructure)을 관리하기에 있어서 유연함과 간단함을 느낄 수 있게 개발되었다. Ruby 와 Earlang으로 개발 되었고 recipe를 작성할 땐 순수 루비 혹은 도메인 특화 언어 (DSL)를 쓸 수 있다.

3.1.1 Open source vs. Enterprise

Features	Open Source Chef	Enterprise Chef
Flexible and Scalable Automation Platform: • Manage 10,000+ Nodes with a Single Chef Server • Easy Installation with 'one-click' Omnibus Installer • Integration with Leading Cloud Providers • Support for Linux, Windows, Mac OS, Solaris, and FreeBSD • Automatic System Discovery and Search Capabilities	✓	✓
Enterprise features including: • Hosted Service • Enhanced Management Console • Multi-Tenancy • Role-Based Access Control • Authentication Using LDAP or Active Directory • High Availability Installation Support and Verification	–	✓
Support and Customer Success: • Standard Support • Support for Test-Driven Infrastructure	–	✓

Table 5. Open Source Chef 와 Enterprise Chef 제공 서비스 비교

무료버전과 유료버전의 차이는 hosted 서비스 와 support를 지원해주느냐 일 것이다. Hosted 서비스 란 Opscode 에서 직접 서버(server)를 생성 및 운영해주는 서비스이다. Chef는 Puppet 과 같이 Server-client 구조를 가졌지만 여기에 Workstation 이 추가된다. 관리자의 작업은 대개 Workstation에서 이루어지기 때문에 서버는 굳이 직접 운영할 필요도 없거니와 보안상의 이유로 hosted server 서비스를 선택하는 유저들이 있다.

3.1.2 Community

Chef 역시 Puppet 처럼 무료버전을 쓰는 유저는 Opscode의 지원을 받을

수 없다. 또한 역시 그러한 유저들을 위해 아래와 같은 다양한 커뮤니티들이 존재한다.

- Mailing list
- Google groups
  - Opscode-Chef-Openstack
  - Chef-supermarket
  - Chef-testing
- Chat channel
  - Chef
  - Chef-hacking
  - Openstack-chef
- Chef 티켓
- Supermarket

Google groups 에는 Openstack 설치 관련 토론이 중심인 Opscode-Chef-Openstack, 유저들이 직접 작성하고 배포하는 Supermarket 에 관련된 프로젝트를 이야기하는 Chef-supermarket, 그리고 여러가지 test 에 대한 포럼인 Chef-testing 이 존재한다. 실시간으로 의견을 주고받을 수 있는 chat channel 역시 다양하게 존재하며 활발하게 돌아가고 있다. 또한 Chef Supermarket 에는 6만명에 가까운 유저들이 1500여가지 이상의 Cookbook 을 공유하고 있다. 기본적이거나 많이 사용되는 어플리케이션(application)들은 이곳에서 받아 설치할 수 있고, 이들이 올린 cookbook 을 참고해 자신만의 cookbook 을 제작하기에 용이하다.

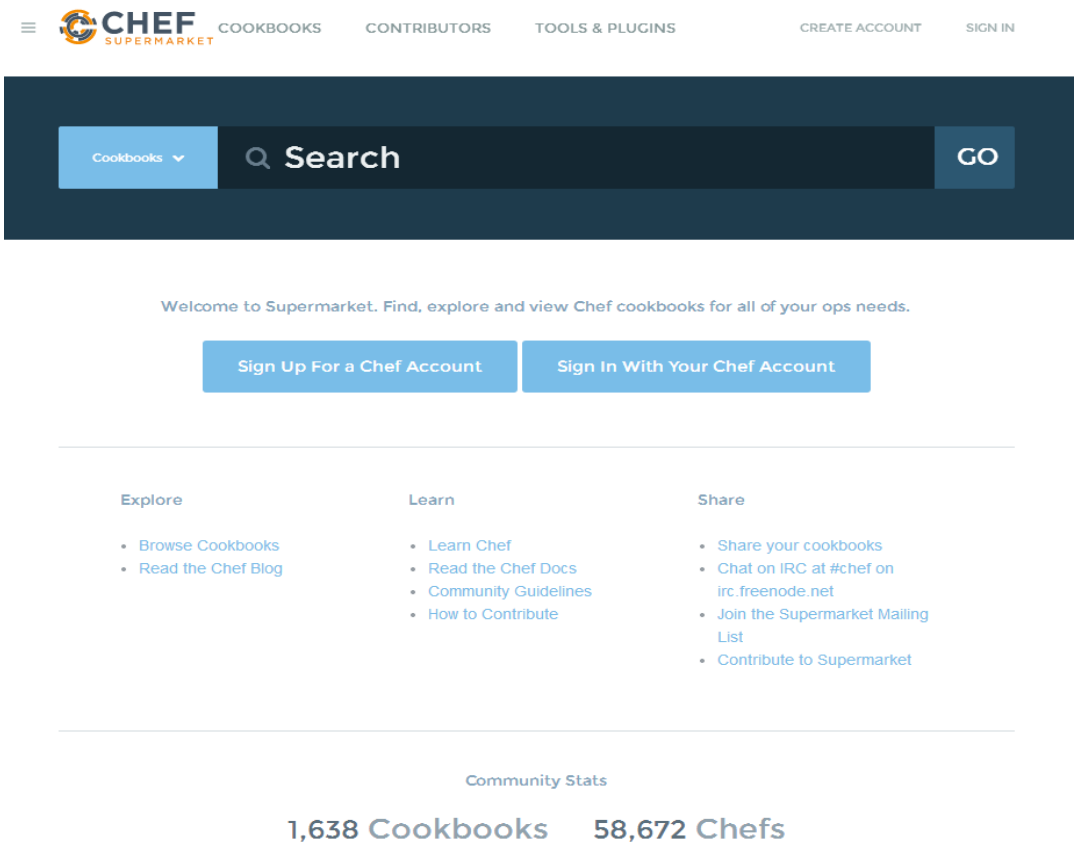


Figure 9. Chef-supermarket 메인 페이지

3.2 Chef 구조

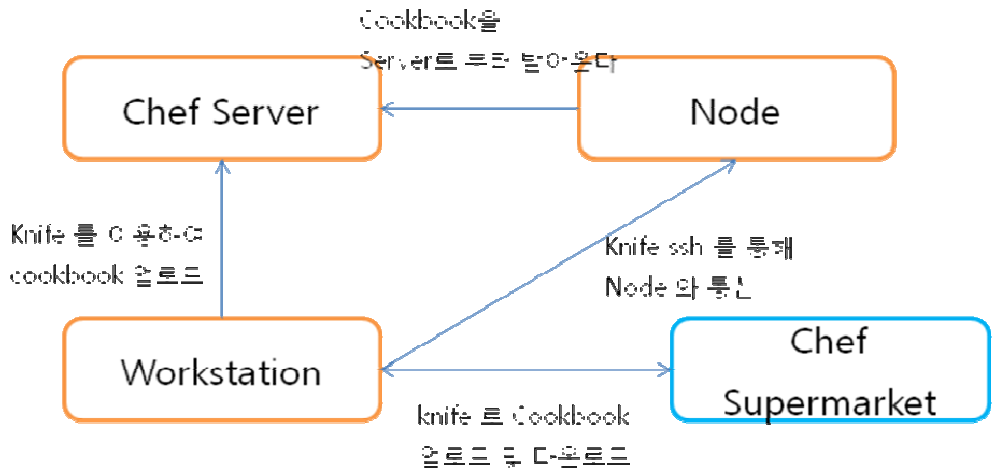


Figure 10. Chef 의 형상관리 구조

Chef 는 Puppet 과 마찬가지로 client-server 구조를 가지고 있다. 하지만 여기에 workstation이 추가되었다. Chef server는 설정 정보가 담긴 Chef 코드 저장소를 가지고 있어 Chef 코드를 집중해서 관리한다. 또한 REST API 를 통해 접근이 가능하며, Web UI 를 통해 Chef server 의 인프라를 관리할 수 있다. Workstation 은 'knife'가 설치된 서버로 주로 Node 에서 사용 가능한 cookbook을 개발하여 서버에 업로드 하는 일을 한다. Chef server랑 비슷해 보여도 Chef server는 어디까지나 Cookbook 을 저장하고 관리 및 배포하는 일종 storage역할을 할 뿐, 직접적인 Cookbook 제작, 수정 및 배포 지시는 Workstation 에서 이루어진다. Node 라고 불리는 client 는 서비스를 구성하기 위한 최소의 물리적 서버이며 Chef server 로부터 run-list 를 가져와 패키지를 설치하고, 설정하며 application 을 실행하기도 한다 [2].

3.3 Puppet 설치

3.3.1 설치 개요

Open source Chef 는 http://www.getchef.com/chef/install/ 에서 여러 환경에 적합한 버전을 다운 받을 수 있다. Puppet 과는 달리 구성해줘야 하는 환경이 3가지 - Chef server, workstation, client - 이다. Chef Solo의 경우 노드는 하나밖에 필요 없지만, 일반 Chef 는 workstation 과 server를

같은 노드안에 설치한다고 해도 노드가 2개 이상 필요하다.

설치 과정은 대략적으로 아래와 같다:

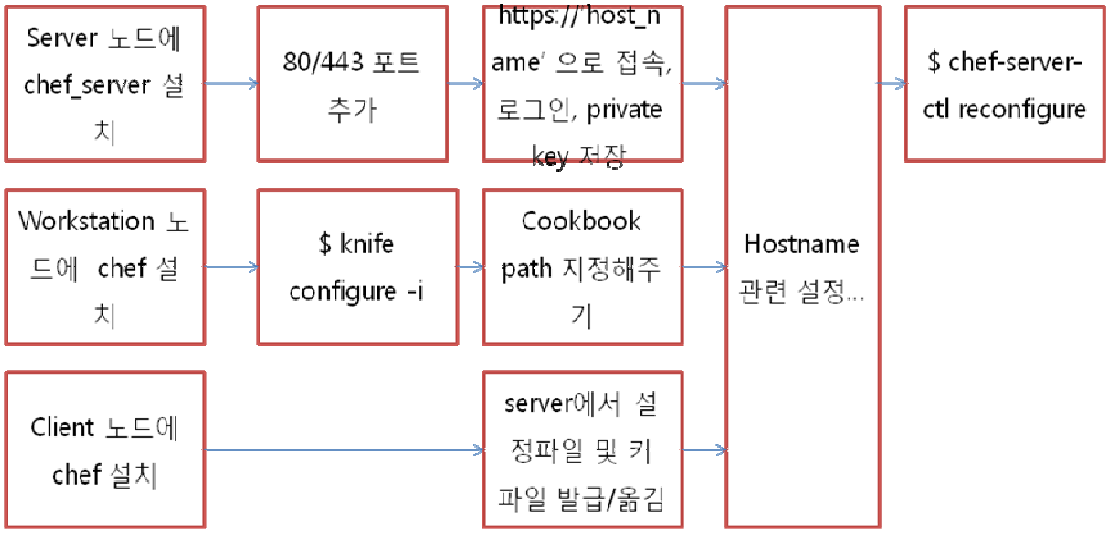


Figure 11. Chef 설치 과정

편의를 위해 server 와 workstation을 같은 노드안에 설치하고 다른 노드 하나에 client를 설치하였다.

	Server / workstation 노드	Client 노드
설치된 OS	Centos 6.5	Centos 6.5
노드 이름 (hostname)	chef-server	node
할당된 ip	172.16.10.10	172.16.10.11

Table 6. Chef Server 와 Client 설치 환경

3.3.2 설치 방법

□ 서버 패키지 설치

```
# wget https://opscode-omnibus-packages.s3.amazonaws.com/
el/6/x86_64/most_recent_chef-server_version.rpm
```

```
# rpm -ihv most_recent_chef-server_version.rpm
```

'most_recent_chef-server_version' 부분에 알맞은 최신 chef-server 버전을 넣어 실행하면 된다. 현재 최신 버전은 chef-server-11.1.4-1.el6.x86_64 이다.

☐ hostname 설정
각각의 노드의 이름을 설정해준다

```
# hostname node_name
```

node_name 대신 노드에 알맞은 이름을 넣어주면 된다.

☐ /etc/hosts 설정

```
Server (workstation)의 경우
172.16.10.6 chef-server
172.16.10.7 node

Client 의 경우
172.16.10.6 chef-server
```

Puppet 과 마찬가지로 server 에는 모든 client 정보를, client 의 경우에는 sever 노드 정보만 있으면 된다.

☐ /etc/sysconfig/network 에서 hostname 설정
각 노드마다 앞에서 설정해준 노드 이름을 입력해준다.

```
# Hostname = node_name
```

☐ /etc/sysconfig/network 에서 hostname 설정

```
# chef-server-ctl reconfigure
```

☐ 80, 443 포트 개방

Chef 는 어떤 종류의 서버 (standalone, high availability, front-end, back-end) 를 이용하냐에 따라 열어줘야 하는 포트 수가 다르다. 하지만 Back-end 서버를 제외하곤 80 과 443 포트만을 열어줘도 충분하다.

☐ Workstation 구축

앞서 언급한 대로 server 와 workstation 은 같은 노드에 구축하려고 한다. Server 를 설치했던 노드에 chef client 를 설치해준다.

```
# curl -L https://www.opscode.com/chef/install.sh | bash
```

server에 접근하기 위해선 비밀키를 서버로부터 복사해와야 한다. Server 와 workstation 이 따로 있으면 sftp 를 통해 가져올 수 있고, 지금 같은 경우 아래 커맨드로 간단하게 복사해주면 된다.

```
# cp /etc/chef-server/chef-validator.pem /etc/chef
```

☐ 새 유저 생성

Knife 명령으로 server 에서 작업하는 사용자를 만든다.

```
# knife configure -i
```

☐ Cookbook 저장 위치 지정

```
cookbook_path Cookbook_Path
```

/root/.chef/knife.rb 에서 위와 같은 항목을 추가해 줌으로써 knife 명령어 실행 시 cookbook 위치를 지정하지 않아도 자동으로 *Cookbook_Path* 에 저장하게 된다. *Cookbook_Path* 대신 cookbook을 저장하고 싶은 디렉토리

위치를 넣어준다.

☐ 설정 파일과 키 생성

Node 가 chef-server에 등록하려면 설정파일(client.rb)과 키(validation.pem)가 필요하다. 아래의 명령어로 /tmp 에다가 설정파일과 키를 생성한다.

```
# knife configure client /tmp
```

☐ Node 구축

Chef-server 에 등록하기 위해 우선 chef client 를 설치한다.

```
# curl -L https://www.opscode.com/chef/install.sh | bash
```

이전에 chef-server(workstation)에서 만든 설정파일과 키를 /etc/chef 로 sftp 를 이용해 옮겨온다.

```
# mkdir -p /etc/chef
# sftp root@172.16.10.10
sftp> get /tmp/client.rb /etc/chef
sftp> get /tmp/validation.pem /etc/chef
```

3.3.3 오류 및 해결법

- **https://'host_name' 으로 접속이 불가능** hostname 바꿔준 후 chef-server-ctl reconfigure 을 실행시켜보자.
- **knife configure -i 실행시 Network Error:connection refused**
  이것 역시 server의 바뀐 설정이 적용 안돼서 나타날 수 있다. chef-server-ctl reconfigure 을 실행시켜주자.
- **ChildConvergeError , No route to host - connect(2)**
  포트 80, 443 을 추가해주거나 iptables 를 내린다.

3.4 Cookbook

Puppet 에 module 이 있다면 Chef 에는 cookbook 이라는 것이 존재한다.

Cookbook – 가장 밖에 있는 디렉토리로 Cookbook 이름과 일치한다.

- recipes : Node를 구축하기 위한 설치 지침을 작성하며 Ruby로 작성된다
- attributes : Recipe 에서 사용할 변수를 기술한다.
- definitions : 기존 Resource 를 결합하여 재사용 가능한 새로운 Resource 를 정의 할 수 있다.
- files : Recipe 에서 사용할 파일이 저장되는 곳이다. 설정 파일과 패키지 파일이 있다.
- libraries : 리소스 기능들을 확장해서 클래스로 만들어 놓으면, 이를 Recipe 에서 호출해서 사용한다.
- templates : File 과 비슷하지만, Attribute 등에 따라 내용을 편집하는 전체의 설정 파일 템플릿. erb 형식으로 작성한다.
- resources : cross-platform 의 추상화가 되어 있어, OS 와 패키지의 버전의 차이를 신경안써도 되게끔 설계되어 있다.
- providers : Resource 를 받고 Resource 와 Node 의 현재 상태를 비교하며 Resource 의 동작을 수행한다. 즉, Resources 의 이행 도구가 된다.
- metadata : Chef 에 Recipe 및 종속성 버전 제약, 지원 플랫폼, Cookbook 간의 의존 관계 등을 전달하는 데 사용한다.

```
├── attributes
│   └── default.rb
├── CHANGELOG.md
├── definitions
├── files
│   └── default
├── libraries
├── metadata.rb
├── providers
├── README.md
├── recipes
│   └── default.rb
├── resources
└── templates
    └── default
        └── ntp.conf.erb
```

Figure 12. ntp cookbook 의 tree 레이아웃

Cookbook 은 어떤 어플리케이션(application)이나 서비스를 설정하기 위한 attributes, recipes, resources, definitions 의 집합체이다. Cookbook 안에는 설치 및 배포에 있어서 필요한 모든 것들이 축약되어있고, 덕분에 설치되는 환경에 따라 수정할 능력만 있으면 얼마든지 다른 유저와 공유가 가능하다. 또한 유저가 직접 cookbook 을 만들어 쓸 수도 있고, knife 명령어를 통해 이미 제작된 것을 내려 받아 쓸 수 있다.

기본적으로 Cookbook 의 작성 순서는:

- Community 에서 필요한 cookbook template 받기
- Attributes 정의
- Templates 파일 생성
- Recipe 작성
- Metadata 작성 및 Cookbook 업로드

3.4.1 Attributes

Attribute은 chef-client 가 자기가 설치되어있는 노드가:

- 현재 어떤 상태(state)인지
- 이전의 chef-client run 으로 어떤 상태가 됐는지
- 이번의 chef-client run 으로 어떤 상태가 되어야하는지

를 이해하기 위해 쓰인다.

```
case platform
  when "ubuntu","debian"
    default[:ntp][:service] = "ntp"
  when "redhat","centos","fedora","scientific"
    default[:ntp][:service] = "ntpd"
end
```

chef-repo/cookbooks/ntp/attributes/default.rb

3.4.2 Templates

Cookbook template 은 Embedded Ruby(ERB) template 으로 template 안의 변수와 논리를 기반으로 파일을 생성하기 위해 쓰인다. Template 을 이용하기 위해선 두 가지 조건이 충족되어야 한다:

- Template 자원을 recipe에 추가
- ERB template 을 cookbook에 추가

예를 들면, Ntp cookbook 의 /recipe/default.rb 에 아래의 자원을 추가한다.

```
template "/etc/ntp.conf" do
  source "ntp.conf.erb"
  owner "root"
  group "root"
  mode 0644
  notifies :restart, "service[ntp]"
end
```

Chef-repo/cookbooks/ntp/recipes/default.rb

그리고는 아래와 같은 ntp.conf.erb 템플릿(template)을 templates/default 디렉토리에 생성해준다.

```
ftfile /var/lib/ntp/ntp.drift

statistics loopstats peerstats clockstats
filegen loopstats file loopstats type day enable
filegen peerstats file peerstats type day enable
filegen clockstats file clockstats type day enable

restrict -4 default kod notrap nomodify nopeer noquery
restrict -6 default kod notrap nomodify nopeer noquery

restrict 127.0.0.1
restrict ::1

<% node[:ntp][:servers].each do |ntpserver| -%>
server <%= ntpserver %>
```

```
<% end -%>
```

Chef-repo/cookbooks/ntp/templates/default/ntp.conf.erb
이렇게 위처럼 template file 과 template resource 설정을 추가해줌으로써 /etc/ntp 의 설정을 관리할 수 있게 되었다.

3.4.3 Recipes

Recipe는 가장 기본적인 설정 요소로:

- Ruby 로 쓰여있다.
- 자원(resource)의 집합.
- 반드시 설정(configuration)에 필요한 모든 것을 정의(define)해야 한다.
- 반드시 Cookbook 안에 존재해야 한다.
- Chef-client 가 사용하기 전에 반드시 run-list에 추가되어야한다.
- 항상 run-list에 나열된 순서대로 실행된다.

Puppet 과는 달리 코드가 작성된 순서대로 실행이 되는데 이는 코드의 가독성과 이해도를 돕는다. 아래는 recipe 코드의 예시다.

```
case node[:platform]
when "ubuntu","debian"
  package "ntpdate" do
    action :install
  end
end

package "ntp" do
  action [:install]
end

template "/etc/ntp.conf" do
  source "ntp.conf.erb"
  owner "root"
  group "root"
```

```
  mode 0644
  notifies :restart, "service[ntp]"
end

service "ntp" do
  service_name node[:ntp][:service]
  action [:enable, :start]
end
```

chef-repo/cookbooks/ntp/recipes/default.rb

제 4장 Heat

4.1 Heat 개요

Heat 은 오픈스택(Openstack)의 오케스트레이션(Orchestration) 프로젝트다. Havana 버전부터 도입되었으며 지속적인 업데이트가 이루어지고 있다. 다만 몇몇 기능은 다음 버전인 Icehouse 만 사용 가능하다. Heat 은 template 이라는 텍스트파일로 여러 개의 클라우드(cloud) application 을 설치 가능하게 하는 heat engine 을 사용하고 있다. Heat template 포맷은 더 쉽고 간편하게 작성 될 수 있도록 보안되고 있다. [3]

Heat 프로젝트만을 위한 공식 커뮤니티는 존재하지 않지만 Openstack 의 ask 사이트나 openstack-dev mailing list 를 통해 정보를 얻을 수 있다. 또한 #heat 또는 #openstack-meeting IRC channel 에서 실시간으로 유저들과 정보를 교환할 수 있다.

4.2 Heat 구조

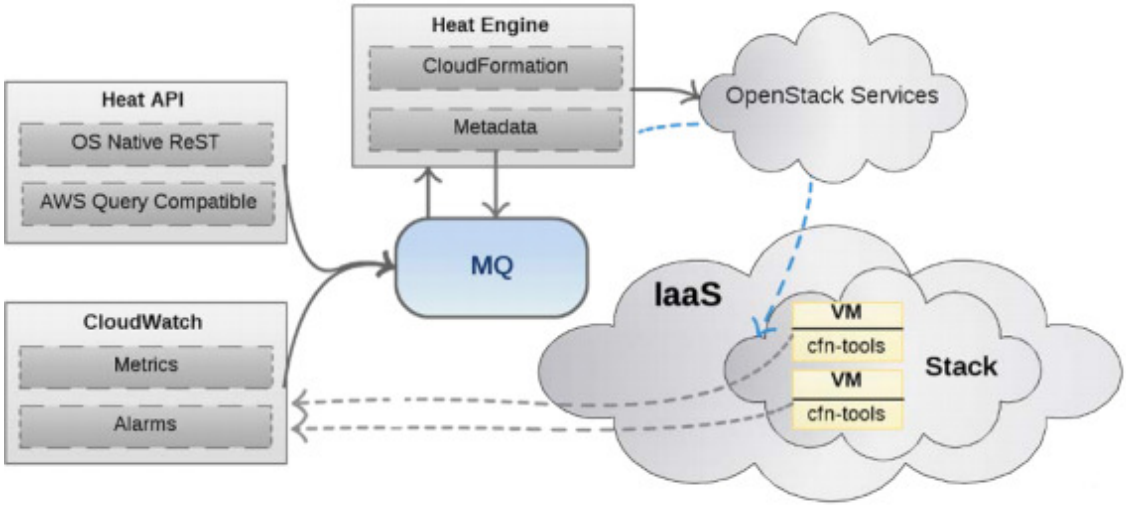


Figure 13. Heat 구조

Heat 은 몇가지 주요 서비스 – Heat engine, Heat API, Heat API-cfn – 로

이루어져있다. **Heat-api** 는 API 요청을 AMQP를 통해 Heat Engine 으로 보내주는 OpenStack-native ReST API를 제공한다. 이를 통해 Heat Engine에게 무엇을 해야 할지 지시한다. **Heat Engine** 은 모든 Orchestration 작업을 도맡아 한다. Template 을 실행시키며 발생한 event를 API 소비자에게 제공해준다. **Heat-api-cfn** 은 AWS CloudFormation 과 호환 가능한 API를 제공해주고 API 요청을 Heat engine으로 전달해준다.

4.3 Heat 설치

4.3.1 설치 개요

Heat 은 Openstack 프로젝트이므로 기본적으로 Openstack 이 설치되어 있어야 한다. 보통 Devstack 이나 Packstack 을 이용해 Openstack 을 설치하기 마련인데 이때 heat 은 설치되지 않으므로 따로 yum 에서 설치 받아야 한다.

설치된 OS	Centos 6.5
Openstack 버전	Havana
Ip 주소	150.183.223.45

Table 7. Heat 설치 환경

4.3.2 설치 방법 [4]

☐ **Heat-api, heat-engine, heat-api-cfn 설치**

```
# yum install openstack-heat-api openstack-heat-engine openstack-heat-api-cfn
```

Yum 을 이용해 heat 모듈을 설치한다.

☐ **location 지정**

오케스트레이션 서비스가 데이터를 저장할곳 데이터베이스의 위치를 지정해 주어야한다. 지금의 경우 MySQL 데이터베이스를 이용할 것이다.

```
# openstack-config --set /etc/heat/heat.conf DEFAULT
sql_connection mysql://heat:HEAT_DBPASS@controller/heat
```

controller 대신 ip주소를, *HEAT_DBPASS* 대신 데이터베이스의 비밀번호를 넣는다.

☐ heat 데이터베이스 생성

MySQL 를 이용해 heat 데이터베이스를 생성해준다.

```
# mysql -u root -p
mysql> CREATE DATABASE heat;
mysql> GRANT ALL PRIVILEGES ON heat.* TO 'heat'@'localhost'
IDENTIFIED BY 'HEAT_DBPASS';
mysql> GRANT ALL PRIVILEGES ON heat.* TO 'heat'@'%'
IDENTIFIED BY 'HEAT_DBPASS';
```

이전에 MySQL 비밀번호를 지정해준 적이 없다면 디폴트 비밀번호는 공백 공백일 것이다. 그러므로 위와 같이 mysql -u root -p 를 입력하면 비밀번호가 맞지 않는다는 에러가 뜨니 대신 mysql -u root 를 입력하자. 위와 마찬가지로 *HEAT_DBPASS* 대신 위에 지정한 비밀번호를 넣어준다.

☐ heat service table 생성

```
# heat-manage db_sync
```

☐ heat 유저 생성

```
# keystone user-create --name=heat --pass=HEAT_PASS --
email=heat@example.com
```

```
# keystone user-role-add --user=heat --tenant=service --
role=admin
```

오케스트레이션 서비스가 identity service 에 인증받을 수 있도록 heat 유저를 생성한다.

☐ /etc/heat/heat.conf 설정

```
[keystone_authtoken]
auth_host = controller
auth_port = 35357
auth_protocol = http
auth_uri = http://controller:5000/v2.0
admin_tenant_name = service
admin_user = heat
admin_password = HEAT_PASS

[ec2_authtoken]
auth_uri = http://controller:5000/v2.0
keystone_ec2_uri = http://controller:5000/v2.0/ec2tokens
```

이전과 마찬가지로 *controller* 와 *HEAT_PASS* 를 알맞게 바꿔 넣어주면 된다.

☐ Identity service 에 등록

```
#keystone service-create --name=heat --type=orchestration --
description="Heat Orchestration API"

# keystone service-create --name=heat-cfn --type=cloudformation -
-description="Heat CloudFormation API"
```

Heat 과 CloudFormation API 를 Identity Service 에 등록함으로써 Openstack 서비스가 이 API들을 찾아낼 수 있다.

☐ endpoint 생성

등록 후 뜨는 id 값들을 이용해 endpoint 를 생성한다.

```
# keystone endpoint-create ₩
--service-id=id1 ₩
--publicurl=http://controller:8004/v1/%₩(tenant_id₩)s ₩
--internalurl=http://controller:8004/v1/%₩(tenant_id₩)s ₩
--adminurl=http://controller:8004/v1/%₩(tenant_id₩)s


# keystone endpoint-create ₩
--service-id=id2 ₩
--publicurl=http://controller:8000/v1 ₩
--internalurl=http://controller:8000/v1 ₩
--adminurl=http://controller:8000/v1
```

id1, id2 에 각각 Heat Orchestration API 와 Heat CloudFormation API 의 id 를 넣으면 된다.

☐ Metadata 와 waitcondition 서버 설정

아래와 같이 /etc/heat/heat.conf 에 meatadata 와 waitcondition 서버 url을 설정해준다.

```
# openstack-config --set /etc/heat/heat.conf DEFAULT
heat_metadata_server_url http://controller:8000
# openstack-config --set /etc/heat/heat.conf DEFAULT
heat_waitcondition_server_url http://controller:8000/v1/waitcondition
```

☐ 8774 포트 개방

☐ heat-api, heat-api-cfn, heat-engine 서비스 실행하기

```
# service openstack-heat-api start
# service openstack-heat-api-cfn start
# service openstack-heat-engine start
# chkconfig openstack-heat-api on
# chkconfig openstack-heat-api-cfn on
# chkconfig openstack-heat-engine on
```

4.4 Heat Template

Heat 은 Puppet 과 Chef 의 Module, Cookbook 처럼 Template 을 이용해 Stack 을 생성한다. Heat Template 은 2 가지 포맷이 존재하는데 HOT 와 AWS CloudFormation Template 이다. HOT 는 Heat Orchestration Template 으로 AWS 와는 달리 오로지 Openstack 에서만 이용 가능한 템플릿이다.

HOT Syntax

```
heat_template_version: 2013-05-23
...
resources:
  ...
  the_resource:
    type: OS::Ceilometer::Alarm
    properties:
      alarm_actions: [Value, Value, ...]
      comparison_operator: String
      description: String
      enabled: Boolean
      evaluation_periods: Integer
      insufficient_data_actions: [Value, Value, ...]
      matching_metadata: {...}
      meter_name: String
      ok_actions: [Value, Value, ...]
      period: Integer
      repeat_actions: Boolean
      statistic: String
      threshold: Number
```

JSON Syntax

```
{
  "AWSTemplateFormatVersion" : "2010-09-09",
  ...
  "Resources" : {
    "TheResource": {
      "Type": "OS::Ceilometer::Alarm",
      "Properties": {
        "alarm_actions": [Value, Value, ...],
        "comparison_operator": String,
        "description": String,
        "enabled": Boolean,
        "evaluation_periods": Integer,
        "insufficient_data_actions": [Value, Value, ...],
        "matching_metadata": {...},
        "meter_name": String,
        "ok_actions": [Value, Value, ...],
        "period": Integer,
        "repeat_actions": Boolean,
        "statistic": String,
        "threshold": Number
      }
    }
  }
}
```

제 5장 비교분석

5.1 개요

지금까지 Puppet, Chef, 그리고 Heat 가 어떤 툴인지를 알아봤다. 같은 자동화 도구지만 작업을 수행하는 방식이나 종류가 다르다는 것을 느낄 수 있었을 것이다. 이 중에서 간편하고 회사의 필요를 잘 충족 시킬 수 있는 툴이 무엇일지 비교분석 하는 것이 이 항목의 핵심이다. 다만 Heat 는 Openstack 프로젝트로 Puppet 과 Chef 둘 다 같이 쓰일 수 있으므로 Heat을 제외한 Puppet 과 Chef 두 도구의 비교에 중심을 두겠다.

회사는 기본적으로 Openstack 이라는 클라우드 컴퓨팅 프로젝트를 이용하며 OS 는 CentOS 6.5 를 쓴다. 최소 100개 이상의 노드를 이용해 클러스터를 생성할 것이며 노드들의 OS는 클라이언트들의 요구에 따라 CentOS외의 것을 이용할 수도 있다.

기본 OS	CentOS 6.5
환경 구성 툴	Openstack
노드 수	100+
사용 목적	슈퍼컴퓨팅(supercomputing) 서비스 제공을 위한 클러스터(cluster) 형성 간소화

5.2 서비스 비교

		Chef	Puppet
Specification Properties			
Specification paradigm	Language	Imperative	Declarative
	User interface	GUI + CLI	CLI +GUI
Abstraction Mechanism	Grouping mechanism	Roles	Classes
	Configuration modules	Cook Books	modules
	Relation Modeling	many-to-many between instances	one -to-many between instances
Deployment Properties			
Scalability	nodes supported	1000-10K	unknown
Work flow	Distributed changes	supported	un supported
Deployment Architecture	Translation agent	central server needed	central server needed
	Distribution mechanism	pull	pull
	Platforms	*BSD, Linux, Mac OS X, Solaris and Windows	*BSD, AIX, Linux, Mac OSX, Solaris
Specification Management Properties			
Usability	Tool as a whole	hard	medium
	Specification testing	multiple environment	multiple environment with dry run
	Monitoring	easy integration with Nagios	reports in metrics with in each node and integration with Nagois
Versioning support		svn or git	svn or git
Specification documentation		comments on code if structured Rdoc can be used	comments on code can generate reference documentation (limited).
Integration with environment		run time discovery	crun time discovery.
Conflict management		modality conflict	unknown.
Workow enforcement		no	no.
Access control		file path based	roles based inside configuration specification.
Support			
Available documentation		extensive reference documentation on web site	extensive reference documentation on web site.
Commercial support		yes	yes.
Community		large and active	large and active.

Puppet 과 Chef 는 기능적인 면에서 큰 차이를 보이진 않는다. 기본적으로 '간단한 template 을 통해 어플리케이션 설치 및 설정, 배포' 라는 기능을 두 도구 다 훌륭히 수행 할 수 있다. 다만 template 을 작성하는데 필요한 언어가 명령형(imperative)인지 선언형(declarative)인지에 따라 유저들이 template 을 작성하는데에 느끼는 어려움이 다를것이다.

Puppet 의 경우 선언형 언어를 쓴다. 선언형 프로그래밍은 프로그램이 어떤 방법으로 해야 하는지를 나타내기 보단 결과적으로 무엇이 되어야 하는지를 설명한다. 그렇기 때문에 유저는 자신이 원하는 '상태'까지 도달하기 위해 필요한 과정들을 굳이 쓰거나 알아야 할 필요가 없다. 반면 Chef 는 명령형 언어들 쓴다. 명령형 프로그래밍은 프로그램의 상태와 상태를 변경시키는 연산을 설명하는 방법으로 컴퓨터가 수행할 명령들을 순서대로 써 놓은 것이다. 그렇기에 어떤 상태에 도달하기까지의 과정을 상세히 알아두어야 하지만 반대로 과정 자체를 자유롭게 변경 시킬 수 있어서 좀 더 유연한 프로그래밍이 가능하다. 그렇기에 상대적으로 Puppet 은 보다 간편히 쓸 수 있는 '관리자'를 위한 도구이고 Chef 는 '개발자'들을 위한 도구라고 생각된다.

5.3 가격 비교

엔터프라이즈의 경우 다른 부가적인 기능 사용도 가능하지만 무엇보다 도구를 개발한 회사에서부터의 지원을 받을 수 있다는 점이 장점이다. 또한 Chef 의 경우 hosted 서비스도 받을 수 있다. 하지만 이러한 서비스들의 대한 대가는 만만치 않다.

# of Nodes	Annual Pricing (dollars)	
	Puppet	Chef
0-5	Free	Free
6-10	Free	1440
21-50	2352-5600	3600
51-99	5712-11088	7200
100-249	10500-26145	-

Table 8. Puppet 과 Chef 의 Enterprise 버전 이용시 연간 드는 비용

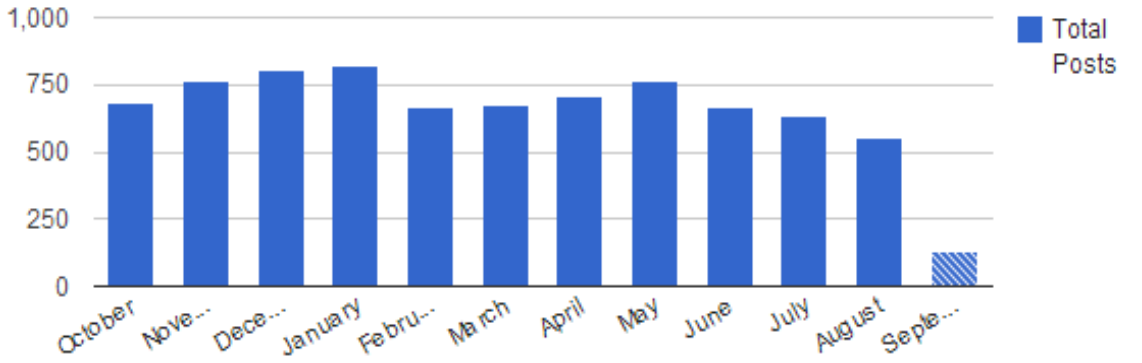
Puppet 과 Chef 엔터프라이즈 버전은 둘 다 맛보기로 5~10 개의 노드를 무료로 사용해 볼 수 있도록 해준다. 다만 이때 회사 지원 서비스 (support service)는 이용해 볼 수 없고 오직 기능적인 면에서만 체험해 볼 수 있다. 무료 체험 가능한 노드 수를 넘어가면 사용 노드 개수만큼 돈을 지불해야 한다. 이때 Puppet 의 경우 노드 각각에 따라 값이 매겨져 있다면 Chef 는 일정한 개수 범위 내에서는 지불할 금액이 일정하다 (e.g. 노드 25 개와 50 개를 이용하는 가격이 같다). 금액 면에선 Chef 가 비교적 저렴하다고 볼 수 있다.

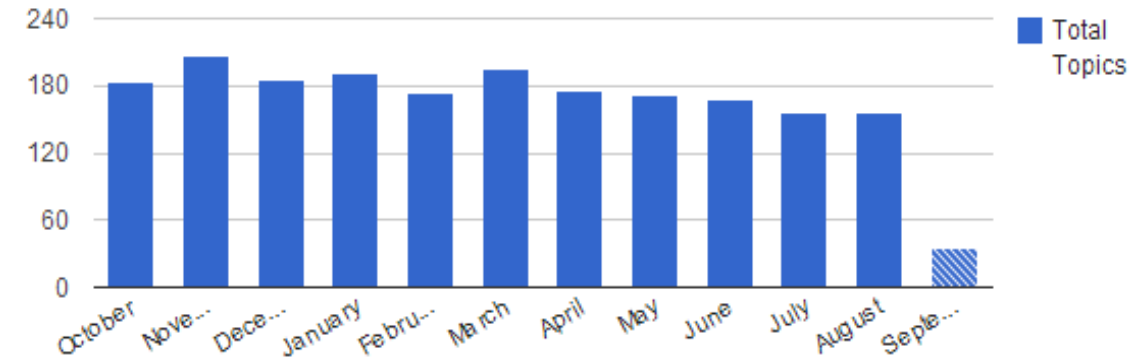
5.4 커뮤니티 비교

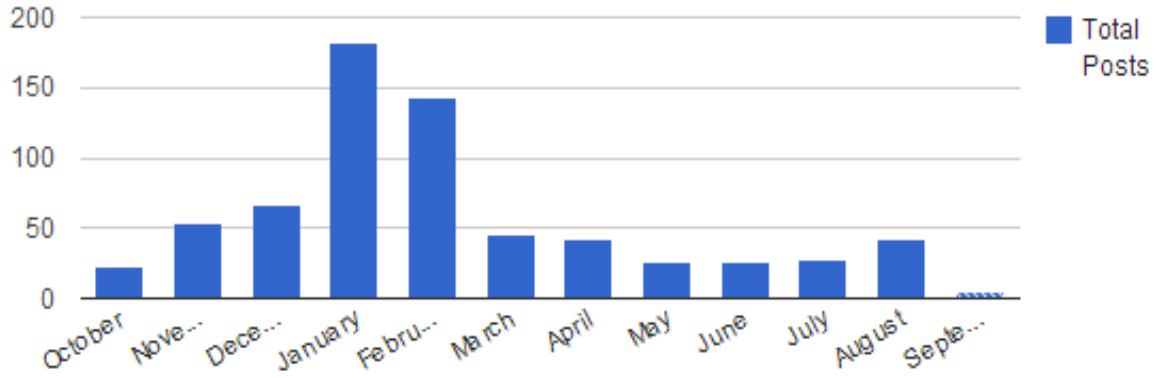
Puppet 은 2005 년에, Chef 는 2009 년에 처음 출시(release)되었다. 그렇기에 선두주자인 Puppet 이 Chef 보다 커뮤니티 측면에서 확고한 기반을 가지고 있다. 아래는 Puppet 과 Chef 의 커뮤니티 크기를 비교한 것이다.

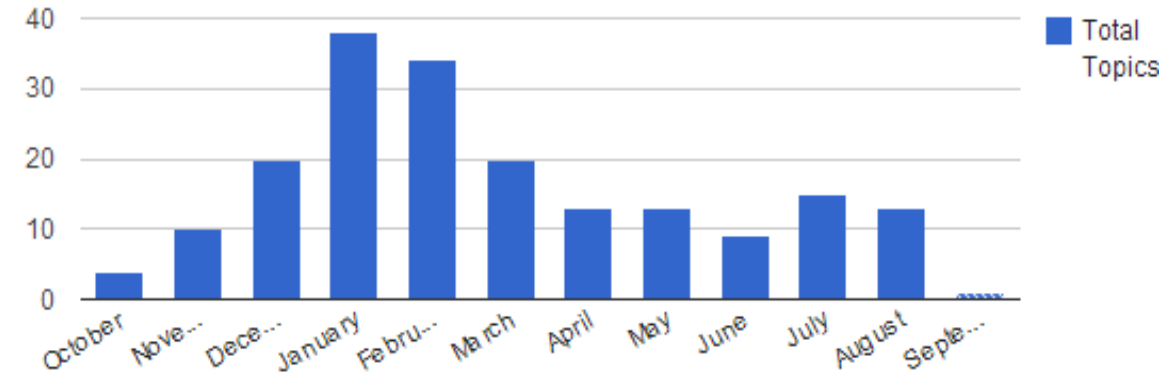
Figure 14. Puppet 과 Chef 관련 커뮤니티

Puppet 과 Chef 둘 다 다양한 구글 그룹스 커뮤니티를 보유하고 있다. 하지만 참여 인원을 보면 Puppet 이 압도적인걸 볼 수 있다.


1,000
750
500
250
0
October
Nove...
Dece...
January
Febru...
March
April
May
June
July
August
Septe...
Total Posts


240
180
120
60
0
October
Nove...
Dece...
January
Febru...
March
April
May
June
July
August
Septe...
Total Topics


200
150
100
50
0
October
Nove...
Dece...
January
Febru...
March
April
May
June
July
August
Septe...
Total Posts


40
30
20
10
0
October
Nove...
Dece...
January
Febru...
March
April
May
June
July
August
Septe...
Total Topics

포스팅 수에서도 차이를 보이는데 위의 통계 그래프에서 Puppet User 그룹은 (좌) 매달 꾸준히 500 여개 이상의 포스트와 120 여개 이상의 토픽이 올라오는 반면 Opscode-chef-openstack 그룹은(우) 1-2 월을 제외하고는 50 여개의 포스트와 10 여개의 토픽만이 올라온다. 이를 통해 커뮤니티의 크기뿐만 아니라 활동빈도 면에서 Puppet 이 우세하다고 볼 수 있다.

포럼 이외에도 공유되는 module 혹은 cookbook 수 역시 Puppet 이 우세하다. 엔터프라이즈 용 유료 module 을 제외하더라도 그 수가 Chef 의 cookbook 의 1.5 배정도 많다.

제 6장 결론

지금까지 Puppet 과 Chef 를 3 가지 – 기능, 가격, 커뮤니티 – 면에서 간단히 비교해보았다. 기능적인 면에서는 더 깊숙이 사용할 경우 발견될 수 있는 차이는 있겠지만 직접 설치해보고 module 혹은 cookbook 을 작성, 배포 해본 결과 언어적인 측면을 제외하고는 두 도구 다 비슷했다. 언어적인 면에서 Puppet 은 원하는 패키지 만을 제시해도 알아서 다운로드 해주는 편리함을 제공하는 동시에 흔히 쓰지 않는 어플리케이션을 받기 위해선 Chef 처럼 구체적인 설명이 필요했다. 배포의 경우 Puppet 과 Chef 둘 다 기본적으로 agent/client 노드에서 pull 하는 형식으로 다운받지만 Chef 는 서버에서 명령어로 push 가 가능했고 Puppet 의 경우 일정 시간 – default 는 30 분– 마다 자동으로 push 하는 기능을 제공해줬다.

엔터프라이즈 버전을 이용할 경우 가격을 무시할 수 없다. Puppet 은 노드마다, Chef 는 일정 노드수 범위 마다 가격을 부여했는데 최대 사용 개수만을 보았을땐 Chef 가 저렴하지만 사용 노드 개수에 따라 Puppet 이 더 저렴할 수도 있다.

커뮤니티 면에선 Puppet 이 압도적으로 우세하다. 4 년 앞선 출시로 더 많은 사용자와 개발자 그룹을 형성하고 있어서 보유 자료와 데이터 양이 Chef 보다 훨씬 많다. 이는 엔터프라이즈가 아닌 오픈 소스를 이용할 때 매우 critical 하다. 오픈 소스의 경우 두 도구 모두 회사에서의 공인된 지원을 받을 수 없다. 그러므로 문제가 생겼을 때는 인터넷 검색으로 직접 문제를 찾고 해결할 수 밖에 없는데, 탄탄한 커뮤니티는 이러한 오픈 소스 유저들에게 큰 도움이 된다.

KISTI 는 최소 100 여개 이상의 노드로 구성된 클러스터에 어플리케이션을 설치하고 설정해줄 수 있는 형상관리 자동화 툴을 요구한다. 엔터프라이즈를 이용할 경우 연 700~1000 만원을 지불해야 된다. 엔터프라이즈를 이용함으로써 디버깅에 드는 시간을 절약할 수 있지만 KISTI 의 인력/인재들을 고려해봤을 때 꼭 지원서비스가 필요하다고 생각하지 않는다. 그러므로 오픈 소스를 이용한다고 가정하자. 위에서 서술한 기능과 커뮤니티를 보았을 때 Chef 보단 Puppet 을 이용하는 것이 더 편리하고 합당하다고 여겨진다. 어지간한 기본적인 어플리케이션은 Puppet Forge 나 GitHub 에서 구할 수 있을뿐더러 활발한 커뮤니티 활동은 문제가 생겼을 시 조언을 구하기에 용이하다. 또한 개인적으로 직접 설치해본 결과 클러스터 환경을 구축하기엔 오픈 소스 Chef 보단 Puppet 이 간단하였다. 그러므로 KISTI 의 슈퍼컴퓨팅을 위한 가상 클러스터 구축에는 형상관리 자동화 도구인 Puppet 이 Chef 보다 더 저렴하고 간편하게 이용할 수 있을 것으로 보인다.