|  |
| --- |
| 리눅스 따라하기 |
| Chef를 통해 시스템 자동 설치하기 |
| Fluentd를 이용한 데이터 수집 자동화 플랫폼 운영 |

|  |
| --- |
| 최두립  2016-03-07 |

[1. Chef 소개 4](#_Toc445230626)

[2.3. 세프에 대하여 4](#_Toc445230627)

[2.4. 세프 아키텍처 4](#_Toc445230628)

[ Chef Server 4](#_Toc445230629)

[ Chef Indexer 5](#_Toc445230630)

[ Node 5](#_Toc445230631)

[ 셰프 클라이언트 5](#_Toc445230632)

[2. Chef 서버 설치 9](#_Toc445230633)

[2.1. Chef 패키지를 다운로드합니다. 9](#_Toc445230634)

[ Chef 싸이트에 접속합니다. 9](#_Toc445230635)

[ 설치할 운영체제를 선택합니다. 9](#_Toc445230636)

[ 운영체제 버전에 맞는 패키지를 선택합니다. 10](#_Toc445230637)

[2.5. Chef 패키지를 설치합니다. 11](#_Toc445230638)

[2.6. chef-server-ctl 을 이용한 서버 설정 11](#_Toc445230639)

[ [관리 콘솔을 실행한다.] 11](#_Toc445230640)

[ [관리자를 계정을 생성한다.] 11](#_Toc445230641)

[ [조직(Organization)을 생성한다.] 12](#_Toc445230642)

[2.7. chef 서비스가 작동 되는지 확인 12](#_Toc445230643)

[ #chef-server-ctl status 12](#_Toc445230644)

[ #chef-server-ctl start 12](#_Toc445230645)

[ #chef-server-ctl status 13](#_Toc445230646)

[ # chef-server-ctl service-list 13](#_Toc445230647)

[2.8. Chef 서버 관리와 부가 기능 13](#_Toc445230648)

[ #chef-server-ctl help 13](#_Toc445230649)

[ chef-server-ctl install package-name 14](#_Toc445230650)

[2.9. 웹브라우저로 관리자 콘솔을 접속해보자 14](#_Toc445230651)

[2.10. Chef 클라이언트 계정 생성 15](#_Toc445230652)

[ 클라이언트를 생성한다. 16](#_Toc445230653)

[ 클라이언트 정보를 입력한다. 16](#_Toc445230654)

[ 등록된 클라이언트의 설정을 확인 한다. 16](#_Toc445230655)

[2.11. CHEF 클라이언트 설치 와 테스트 17](#_Toc445230656)

[2.12. 참고 할 기타 설치 방법 18](#_Toc445230657)

[3. Chef Workstatuion 설정 18](#_Toc445230658)

[3. Knife 설정 18](#_Toc445230659)

[2.1 웹콘솔로 접속하여 계정 등록과 인증서를 생성한다. 18](#_Toc445230660)

[ Reset Validation Key 19](#_Toc445230661)

[ Generate Knife Config 19](#_Toc445230662)

[3.3. 사용자 계정의 .chef로 파일을 저장한다. 20](#_Toc445230663)

[ chefadmin.pem생성 20](#_Toc445230664)

[ Knife.rb 20](#_Toc445230665)

[3.4. knife가 제대로 작동하는지 테스트 한다. 21](#_Toc445230666)

[ knife ssl check 21](#_Toc445230667)

[ knife ssl fetch 21](#_Toc445230668)

[ $ knife ssl check 21](#_Toc445230669)

[ $ knife client list 21](#_Toc445230670)

[ $ knife user list 21](#_Toc445230671)

[ $ knife client list 22](#_Toc445230672)

[3.5. knife 환경 설정 하는 다른 방법 22](#_Toc445230673)

[ knife configure –i 를 실행 하여, 다음과 같이 설정 과정을 진행한다. 22](#_Toc445230674)

[Chef Node 초기 설정 26](#_Toc445230675)

[2.1 ssh 설정 26](#_Toc445230676)

[Chef 서버 설치하고 테스트해보기 26](#_Toc445230677)

[3.6. vagrant + virtualbox로 5대의 멀티 인스턴스 올리기 27](#_Toc445230678)

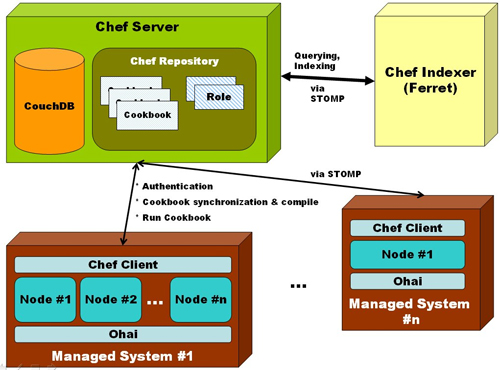
# Chef 소개

## 세프에 대하여

## 세프 아키텍처

#### 프레임워크를 이용할 때 단순히 채워야 할 부분만 채워 애플리케이션을 구성할 수도 있지만 이해 없이 시키는 것만 해서는 멀고 높게 가기 힘들다. 셰프에도 이 원리는 그대로 적용된다. 따라서 셰프를 구축하고 이용하기에 앞서 셰프의 아키텍처적인 면과 각 부분을 이루는 요소에 대해 알아보자. 먼저 셰프의 아키텍처를 개략적으로 살펴보면 다음과 같다.

#### 셰프 환경은 셰프 서버와 셰프 인덱서(indexer), 셰프 클라이언트 로 구성된다



### Chef Server

셰프 서버 운영, 관리 또한 자동화할 수 있다. 노드 정보를 저장하기 위해 CouchDB([http://couchdb.apache.org](http://couchdb.apache.org/))를 사용한다. 노드 인증이나 설정 등 노드 관리와 쿡북을 각 노드에 배치하는 작업 그리고 인덱스 기능 사용 등 기능을 제공한다.

### Chef Indexer

인덱스 서버와 셰프 서버는 메시지 큐를 통해 서로 통신을 수행하며, STOMP([http://stomp.codehaus.org](http://stomp.codehaus.org/))를 이용한다. 사용자는 텍스트 검색(full-text search) 엔진을 통해 현재 셰프 환경 내에 존재하는 것을 검색 할 수 있다.

### Node

셰프에서 노드는 레시피(recipe)가 적용될 시스템을 말한다. 노드에서 셰프 클라이언트 최초 실행 시 셰프 클라이언트는 해당 노드의 OpenID를 셰프 서버에 등록하고, 관리자가 셰프 서버에서 해당 OpenID의 등록을 완료(validate)하면 노드 설정이 마무리된다. 셰프 서버에서 설정된 노드에 레시피나 역할을 할당하거나 제거할 수 있다.

### 셰프 클라이언트

셰프 클라이언트는 셰프 서버와 STOMP를 이용해 통신을 수행하고 노드 인증을 담당한다. 또한 서버와 노드의 쿡북을 동기화하고 새로 받은 쿡북을 컴파일하여 노드에서 실행하는 역할도 수행한다. 셰프 클라이언트는 시스템당 하나만 있으면 해당 시스템의 모든 노드를 관리할 수 있다. 셰프 클라이언트는 필요할 때마다 chef-client 명령을 통해 실행할 수도 있고, 백그라운드에서 주기적으로 변경 사항을 자동 검사하도록 할 수도 있다.

#### Chookbook

한 개 이상의 레시피, 정의, 자원, 속성, 파일, 라이브러리, 템플릿, 메타데이터를 묶어놓은 것이다. 셰프에서 쿡북은 배포의 기본 단위다.

쿡북은 셰프 저장소의 cookbooks 디렉터리 내부에 서브 디렉터리로 존재하며 각 쿡북 디렉터리는 아래와 같이 구성된다.

attributes/

definitions/

files/

libraries/

metadata.rb

README.rdoc

recipes/

templates/

#### 자원(resource)

셰프 내에서 작업의 기본 단위다. 패키지나 파일, 디렉터리, 서비스, 심지어 크론이나 마운트도 자원으로 다룰 수 있다. 자원에 지정된 동작(action)은 적용 시점에 각 노드에서 실행된다. 이를 통해 사용자나 그룹을 생성, 삭제하거나 노드 내 서비스(웹, 데이터베이스)를 재시작할 수도 있으며, 로컬 파티션이나 원격 디렉터리를 마운트한 후 사용할 수도 있다. 현재 기본으로 제공되는 자원은 Cron, Directory, File, HTTP Request, Package, Script, Service 등 총 18가지다.

#### 정의(definition)

기존 자원을 이용해 사용자 고유의 자원을 만들어 사용할 수 있다.

#### 레시피(recipe)

레시피에는 한 개 이상의 자원이 어떤 순서로 관리되는지를 정의한다. 레시피는 다른 레시피를 포함할 수 있고, 쿡북에 정의된 속성뿐만 아니라 레시피를 적용하는 대상 노드에 정의된 모든 속성 역시 이용할 수 있다.

#### 프로바이더(provider)

자원에 지정된 동작을 실제로 수행하는 주체다. 예를 들어 다음과 같이 디렉터리 자원을 정의했다고 하자.

|  |
| --- |
| directory "/tmp/monkey" do  owner "root"  group "root"  mode 0755  action :create  end |

해당 자원이 노드에 적용되면 셰프는 directory 자원을 담당할 프로바이더를 찾는다. 디렉터리는 Chef::Provider::Directory가 담당하며, 이 프로바이더는 /tmp/monkey의 상태를 확인하고 존재하지 않는다면 새로운 디렉터리를 만든다.

#### 라이브러리(library)

라이브러리는 반복적이거나 레시피에 기술하기에는 너무 길고 한눈에 알아보기 힘든 내용을 레시피 외부에 정의하고 레시피에서 사용할 수 있도록 해준다. 라이브러리는 루비 코드나 셰프의 DSL을 확장하는 형태로 작성 가능하다. 각 쿡북의 libraries 디렉터리에 위치하며, 존재하는 라이브러리 파일은 실행 시 자동으로 로딩되고 레시피나 속성, 정의에서 사용 가능하다.

#### 속성(attribute)

레시피 적용 시 노드에서 사용될 값을 지정할 수 있다. 속성이 중요한 이유는 대상 OS마다 다르게 지정해야 할 값을 레시피에 일일이 기술하거나 각각을 별도 파일로 만들 필요가 없게 해주기 때문이다. 아파치 서버를 예로 들면 데비안, 우분투 계열과 레드햇 계열 리눅스는 기본적으로 설치 시에 정해지는 아파치 설정 파일 경로나 이름 등이 좀 다르다. 이를 레시피에 일일이 기술하거나 여러 종류의 레시피를 만든다면 내용도 쓸데 없이 길어지고 복잡도만 늘어난다. 속성을 이용해 이 문제를 쉽게 해결할 수 있다. 아래 예제를 보면 어느 정도 감이 올 것이다.

|  |
| --- |
| Case platform  when "redhat","centos","fedora","suse"  set[:apache][:dir] = "/etc/httpd"  set[:apache][:log\_dir] = "/var/log/httpd"  set[:apache][:user] = "apache"  set[:apache][:binary] = "/usr/sbin/httpd"  when "debian","ubuntu"  set[:apache][:dir] = "/etc/apache2"  set[:apache][:log\_dir] = "/var/log/apache2"  set[:apache][:user] = "www-data"  set[:apache][:binary] = "/usr/sbin/apache2"  else  set[:apache][:dir] = "/etc/apache2"  set[:apache][:log\_dir] = "/var/log/apache2"  set[:apache][:user] = "www-data"  set[:apache][:binary] = "/usr/sbin/apache2"  end |

#### 파일(file)

모든 것이 작성하는 코드만을 이용해 해결된다면 세상은 정말 행복할 것이다. 하지만 실제로는 종종 필요한 파일을 직접 업로드해야 할 경우가 발생한다. 예를 들어 각 리눅스/유닉스 시스템의 패키지 관리자가 지원하지 않는 파일을 설치해야 한다거나, 직접 제작한 파일을 각 시스템에 배포해야 하는 경우가 그러하다.

#### 템플릿(template)

파일과 매우 비슷하지만 한 가지 결정적인 다른 점은 노드에 템플릿이 전송될 때 렌더링 작업을 거친 후 전송된다는 것이다. 각 노드의 시스템 속성 및 정의된 속성이 템플릿 렌더링 시 사용된다. 이것이 의미하는 바는 앞서 속성에서 언급한 바와 같이 각 노드 별로 달라야 하는 값 때문에 설정 파일을 여러 개 만들 필요가 없다는 것이다. 템플릿 렌더링에는 Erubis(<http://www.kuwata-lab.com/erubis>), 즉 ERB 파일이 이용된다. ERB 파일은 머브나 레일스, 또 다른 루비 기반 시스템 관리 소프트웨어인 puppet(<http://reductivelabs.com/trac/puppet>)에서도 사용된다.

#### 역할(role)

노드에 적용될 여러 레시피와 속성의 집합이다. 노드 하나에 하나 이상의 역할을 할당할 수 있다. 역할은 레시피나 속성을 역할의 특성에 맞게 쉽게 조합 가능하며, 역할 이름이 무엇을 하는지 명확히 나타내주기 때문에 차후 유지보수에도 도움이 된다. 예를 들어 필자의 회사에서 구축한 시스템은 여러 파트로 나뉘는데 이미지 프로세싱 서버는 내부적으로 아파치 ActiveMQ 서버와 직접 개발한 루비 데몬, 스카이넷(Skynet)이라는 분산 프로그래밍용 프레임워크가 실행된다. 이때 ActiveMQ, Worker, 스카이넷을 담당하는 레시피를 만든 후 필요한 노드에 각각 할당할 수도 있지만, 세 레시피를 ImageProcessingServer라는 역할로 묶은 뒤 그 역할을 필요한 노드에 할당할 수도 있다. 후자 쪽이 나중에 새 시스템이 추가되었을 때 무엇을 할당해야 할지 더욱 명확하게 해주고, 필요한 레시피를 실수로 누락하지 않도록 해준다.

#### 메타데이터(metadata)

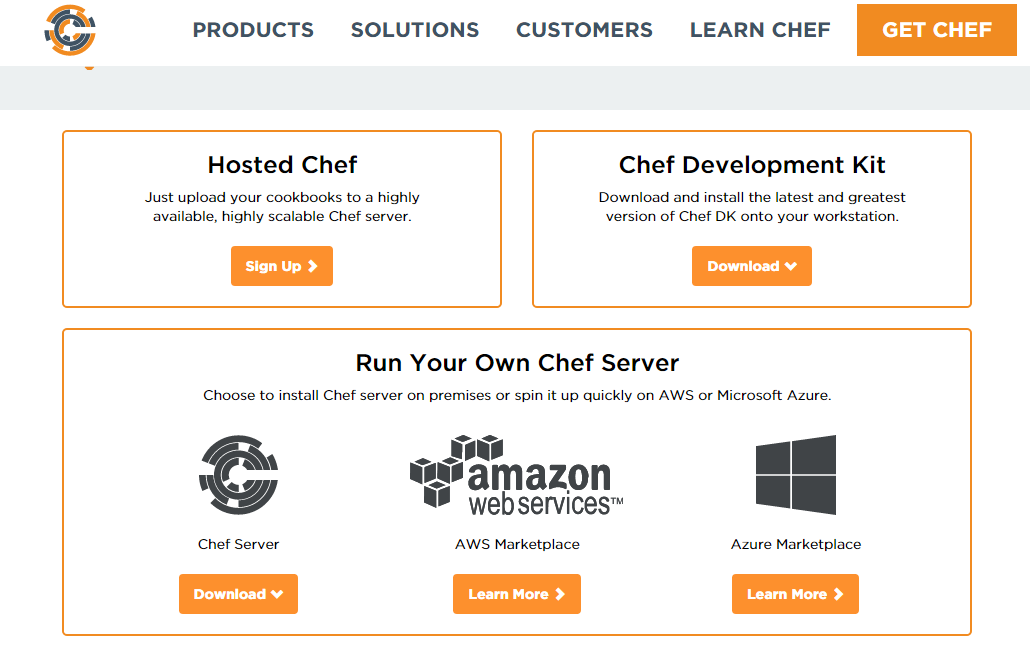
셰프 서버가 각 노드에 제공하는 몇 가지 기본적인 정보를 담고 있다. 각 쿡북의 최상위 디렉터리에 metadata.rb 파일을 만들고 필요한 내용을 기술하면 해당 파일은 JSON파일로 컴파일된 후 셰프 환경에서 사용된다.

# Chef 서버 설치

## Chef 패키지를 다운로드합니다.

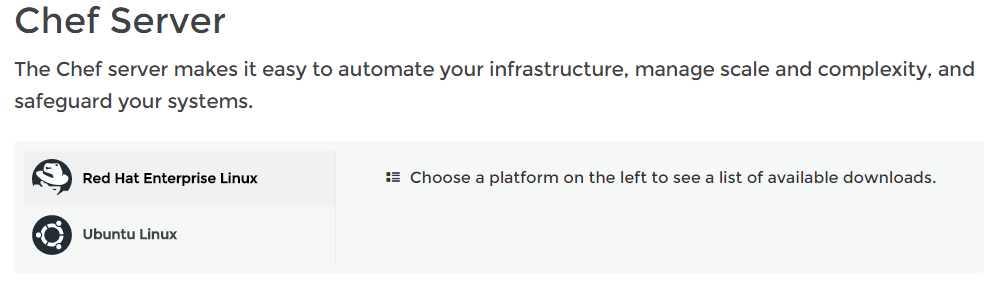
### Chef 싸이트에 접속합니다.

<https://www.chef.io/chef/> 또는 [www.getchef.com](http://www.getchef.com/) 접속하여 “GET CHEF”로 패키지를   
다운받습니다.



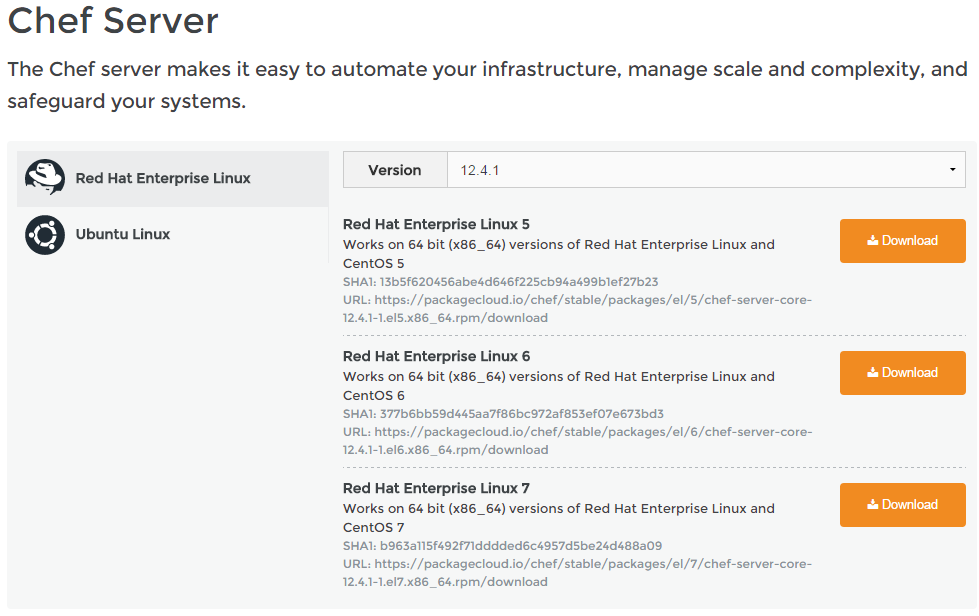
### 설치할 운영체제를 선택합니다.

Chef 서버가 지원하는 운영 체제는 RedHat 또는 Ubuntu Linux입니다. 해당하는 운영체제를 선택합니다.



### 운영체제 버전에 맞는 패키지를 선택합니다.

RedHat Linux6 또는 Centos6 버전에 설치 한다면 “Red Hat Enterprise Linux 6” 를 선택하여 다운 받습니다. 다운로드를 위한 정보 입력을 하고 전송하면 다운로드를 시작합니다.



다운 받는 패키지는 약 450MB가 넘기 때문에 저장공간과 다운로드 시간을 고려해야 합니다.

저는 다른 리눅스 서버에서 Wget으로 다음과 같이 입력하여 다운 받았습니다.

[get\_chef.sh] 라는 스크립트를 만들어 실행하면 됩니다.

**[root@dev1 chef-src]# cat get\_chef.sh**

REFER="https://downloads.chef.io/chef-server/redhat/"

CHEF6="https://packagecloud.io/chef/stable/packages/el/6/chef-server-core-12.4.1-1.el6.x86\_64.rpm/download"

CHEF7="https://packagecloud.io/chef/stable/packages/el/7/chef-server-core-12.4.1-1.el7.x86\_64.rpm/download"

wget "$CHEF6" --ref="$REFER" -O chef-server-core-12.4.1-1.el6.x86\_64.rpm &

wget "$CHEF7" --ref="$REFER" -O chef-server-core-12.4.1-1.el7.x86\_64.rpm &

**[root@dev1 chef-src]# sh get\_chef.sh**

## Chef 패키지를 설치합니다.

다운 받은 패키지를 rpm 명령을 통해 설치합니다. 일반적으로 rpm 설치 옵션은 –Uvh -ivh 를 활용하여 설치합니다.

rpm -Uvh chef-server-core-12.4.1-1.el6.x86\_64.rpm

## ****chef-server-ctl**** 을 이용한 서버 설정

### [관리 콘솔을 실행한다.]

# chef-server-ctl reconfigure

관리 콘솔을 실행 하게 되면 postgres와 opscode-expander 등이 구동된다.

실행 되는 서버 환경은 /var/opt/opscode에 존재 하며 다음 명령을 통해 구동하고 중지가 가능하지만 중지는 잘되지 않았다.

### [관리자를 계정을 생성한다.]

# chef-server-ctl **user-create** user\_name first\_name last\_name email password --filename FILE\_NAME

example)

# chef-server-ctl user-list

pivotal

# chef-server-ctl **user-create** chefadmin admin admin couplewith@yahoo.co.kr 1234567 --filena

me /etc/chef/users/chefadmin.key

# chef-server-ctl **user-list**

chefadmin

pivotal

### [조직(Organization)을 생성한다.]

# chef-server-ctl **org-create** short\_name full\_organization\_name --association\_user user\_name --filename FILE\_NAME

example)

# chef-server-ctl **org-create** cheforg dev.im.com –association\_user chefadmin –filename /etc/chef/users/cheforg.key

# chef-server-ctl **org-list**

cheforg

## chef 서비스가 작동 되는지 확인

### #chef-server-ctl status

down: bookshelf: 60504s, normally up; run: log: (pid 18334) 90405s

down: nginx: 60503s, normally up; run: log: (pid 18493) 90400s

down: oc\_bifrost: 60503s, normally up; run: log: (pid 18114) 90413s

down: oc\_id: 60502s, normally up; run: log: (pid 18127) 90412s

down: opscode-erchef: 60502s, normally up; run: log: (pid 18377) 90404s

down: opscode-expander: 60500s, normally up; run: log: (pid 18265) 90406s

down: opscode-solr4: 60499s, normally up; run: log: (pid 18216) 90408s

down: postgresql: 60498s, normally up; run: log: (pid 18069) 90414s

down: rabbitmq: 60498s, normally up; run: log: (pid 17937) 90418s

down: redis\_lb: 60498s, normally up; run: log: (pid 18488) 90400s

### #chef-server-ctl start

[root@dev1 opscode]# chef-server-ctl start

ok: run: bookshelf: (pid 21092) 1s

ok: run: nginx: (pid 21133) 0s

ok: run: oc\_bifrost: (pid 21139) 1s

ok: run: oc\_id: (pid 21174) 0s

ok: run: opscode-chef-mover: (pid 21177) 0s

ok: run: opscode-erchef: (pid 21208) 1s

ok: run: opscode-expander: (pid 21213) 0s

ok: run: opscode-solr4: (pid 21218) 0s

ok: run: postgresql: (pid 21221) 1s

ok: run: rabbitmq: (pid 21228) 0s

ok: run: redis\_lb: (pid 21266) 1s

### #chef-server-ctl status

[root@dev1 opscode]# chef-server-ctl status

run: bookshelf: (pid 21092) 17s; run: log: (pid 18334) 90434s

run: nginx: (pid 21133) 16s; run: log: (pid 18493) 90429s

run: oc\_bifrost: (pid 21139) 16s; run: log: (pid 18114) 90442s

run: oc\_id: (pid 21174) 15s; run: log: (pid 18127) 90441s

run: opscode-erchef: (pid 21208) 15s; run: log: (pid 18377) 90433s

run: opscode-expander: (pid 21213) 14s; run: log: (pid 18265) 90435s

run: opscode-solr4: (pid 21218) 14s; run: log: (pid 18216) 90437s

run: postgresql: (pid 21221) 14s; run: log: (pid 18069) 90443s

run: rabbitmq: (pid 21228) 13s; run: log: (pid 17937) 90447s

run: redis\_lb: (pid 21266) 13s; run: log: (pid 18488) 90429s

### # chef-server-ctl service-list

bookshelf\*

nginx\*

oc\_bifrost\*

oc\_id\*

opscode-chef-mover\*

opscode-erchef\*

opscode-expander\*

opscode-solr4\*

postgresql\*

rabbitmq\*

redis\_lb\*

## Chef 서버 관리와 부가 기능

chef-server-ctl은 관리자를 위한 다양한 기능을 포함하고 있는데, 그 중 외부 패키지를 설치하기 위한 **install**기능을 가지고 있다

### #chef-server-ctl help

Help 명령으로 다양한 관리가 가능하다.

### chef-server-ctl install package-name

You must supply an addon name. Valid names include: chef-ha, chef-sync, chef-manage, opscode-push-jobs-server, opscode-reporting.

1) chef-manage를 설치한다.

# chef-server-ctl install chef-manage

2) 관리 툴을 설치하고 관리툴의 환경을 설정한다.

# chef-manage-ctl reconfigure

chef-manage Reconfigured!

3) 다음으로 실행한다.

#chef-server-ctl reconfigure

\* 각 툴들은 다음과 같은 과정의 순서로 설치하면 된다.

# chef-server-ctl install opscode-manage

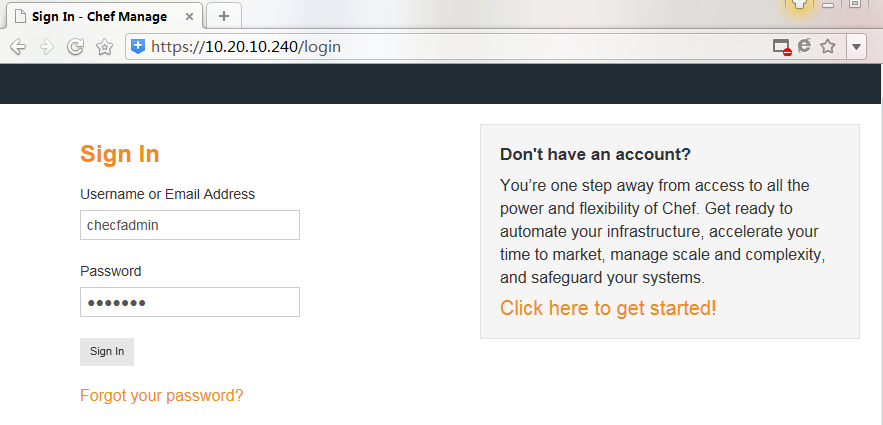
# opscode-manage-ctl reconfigure

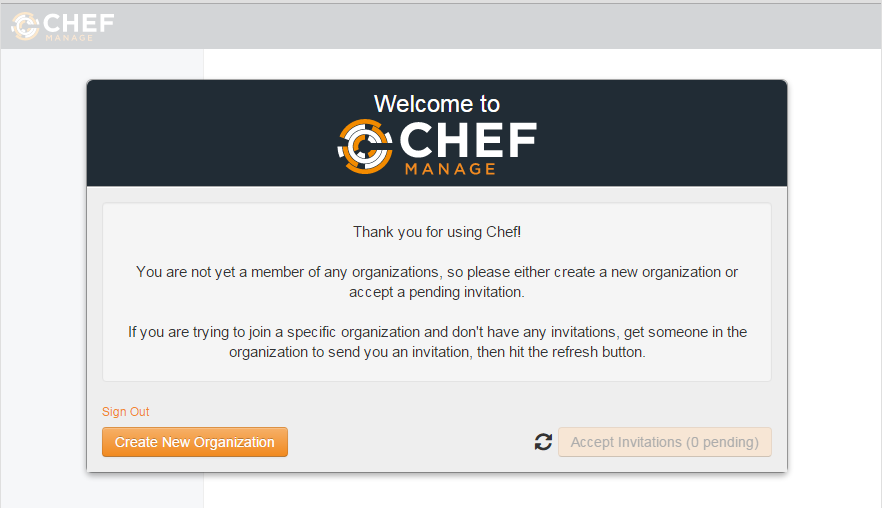
#chef-server-ctl reconfigure

대표적으로 사용하는 chef-ha, chef-sync, chef-manage, opscode-push-jobs-server, opscode-reporting 등과 같은 관리 패키지들을 관리 할 수 있다.이 명령은[http://packagecloud.io](http://packagecloud.io/)의 패키지들을 다운로드하여 설치한다.

## 웹브라우저로 관리자 콘솔을 접속해보자

서버의 주소로 <http://dev1.im.com>으로 접속을 하고 등록한 사용자 chefadmin / 1234567로 로그인이 되는지 확인한다.



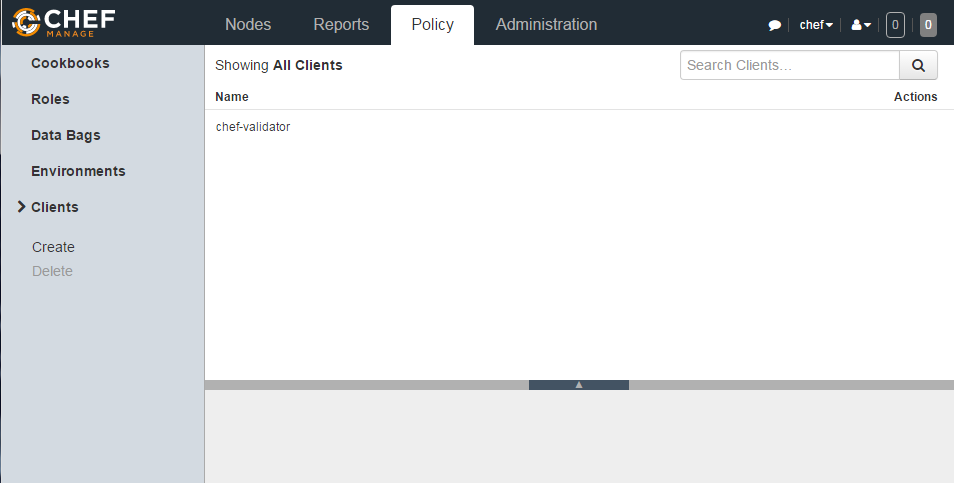


## Chef 클라이언트 계정 생성

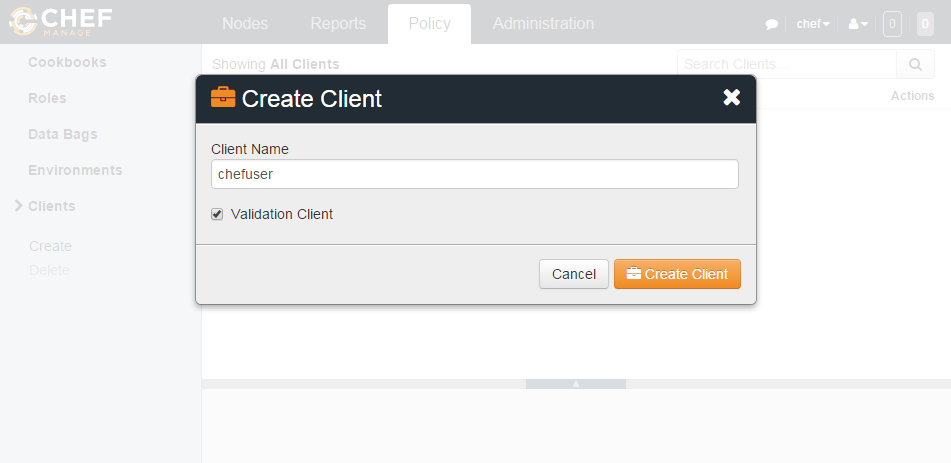
Chef server에 접근해서 데이터 작업을 하려면, chef에 클라이언트를 등록해야 한다.

웹콘솔에 로그인 한 후 **Policy > Clients > Create**를 선택하여 생성하면 된다.

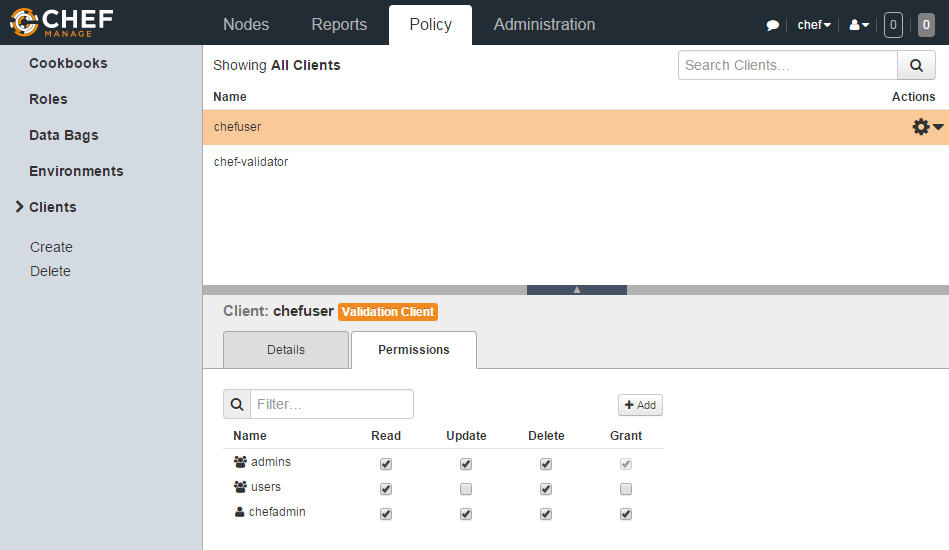
### 클라이언트를 생성한다.



### 클라이언트 정보를 입력한다.



### 등록된 클라이언트의 설정을 확인 한다.



클라이언트 등록시 서버와 인증에 사용할 RSA Private key를 만들어준다. Private key는 작업 PC에 다운로드 받아 저장해 두어야 한다.

## CHEF 클라이언트 설치 와 테스트

solo, knife, ohai 등과 같은 chef client 툴로 작업을 하기 위해 클라이언트 패키지도 설치 해야 한다. 다음과 예와 같이 wget으로 파일을 다운 받아 설치할 수 있다.

설치 확인은 chef-client –v 로 버전을 확인하면 된다.

wegt [**https://opscode-omnibus-packages.s3.amazonaws.com/el/6/x86\_64/chef-12.4.1-1.el6.x86\_64.rpm**](https://opscode-omnibus-packages.s3.amazonaws.com/el/6/x86_64/chef-12.4.1-1.el6.x86_64.rpm)

[root@dev1 chef-src]#rpm –Uvh chef-12.4.1-1.el6.x86\_64.rpm

[root@dev1 chef-src]#rpm –ql chef | grep "/bin"

/opt/chef/bin/chef-apply

/opt/chef/bin/chef-client

/opt/chef/bin/chef-service-manager

/opt/chef/bin/chef-shell

/opt/chef/bin/chef-solo

/opt/chef/bin/chef-windows-service

/opt/chef/bin/knife

/opt/chef/bin/ohai

[root@dev1 chef-src]# chef-client -v

Chef: 12.4.1

## 참고 할 기타 설치 방법

Chef 서버와 클라이언트 설치는 상황에 따라 다음과 같은 다른 설치 방법을 사용 할 수도 있다.

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

# rpm –ivh <https://opscode-omnibus-packages.s3.amazonaws.com/el/6/x86_64/chef-12.4.1-1.el6.x86_64.rpm>

# Chef Workstatuion 설정

chef workstation은 chef 개발자가 작업하는 공간 즉 개발자 PC라고 생각하면된다. 개발자는 workstation에 접속해서 chef를 통해서 resource를 관리하게 된다.

이때 가장 많이 사용하는 knife이다. Knife를 통해서 cookbook, Data bags, chef node, role 등을 관리하게 된다.

knife로 할 수 있는 일들은 다음과 같다.

Cookbook의 생성,삭제,수정

Data Bags의 생성, 삭제, 수정

Chef Node의 등록, 삭제, 수정

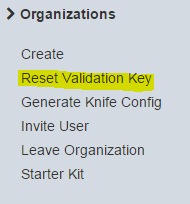
Role의 등록, 삭제, 수정

# Knife 설정

Knife를 chef를 접근하기위해 chef client 등록하고 해당 조직을 생성해야 한다.

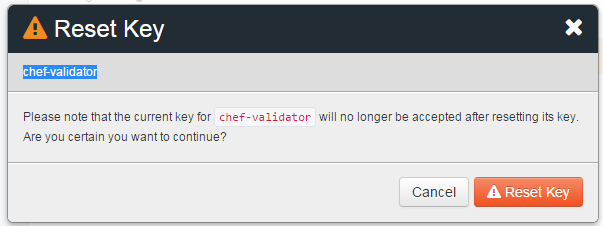
앞에서 생성한 chefuser chef 조직을 이용하기로 한다. 앞서 클라이언트 등록시 생성된 인증서를 chefuser.pem으로 사용한다.

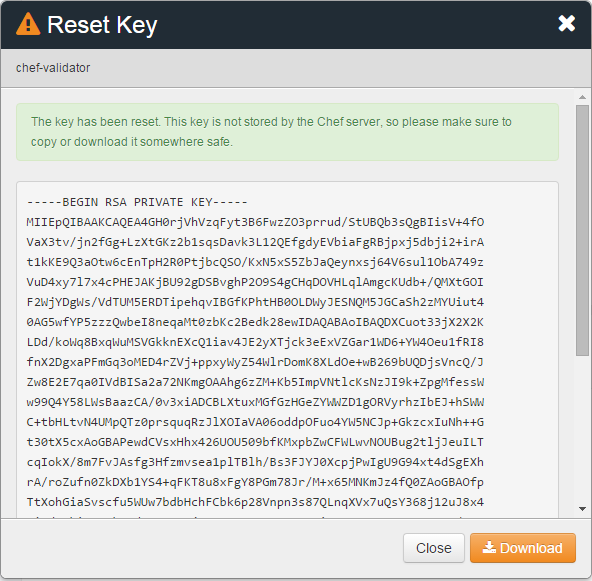
## 웹콘솔로 접속하여 계정 등록과 인증서를 생성한다.

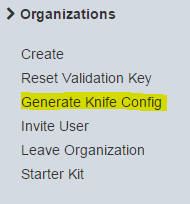


### ****Reset Validation Key****

웹 콘솔의 **Administration > Organizations > Reset Validation Key**명령을 이용해서 chef**-validator.pemdmf 사용자 컴퓨터에** 저장한다.







### Generate Knife Config

**Administration > Organization > Generate Knife Config**를 이용해서 knife.rb설정 파일을 다운 받는다.

현재까지의 작업으로 3개의 파일이 준비되었다.

knife.rb

chefuser.pem

chef-validator.pem

## 사용자 계정의 .chef로 파일을 저장한다.

이제 이들 파일을 저장하기 위한 .chef 디렉토리를 만들어서 위 3개의 파일을 모두 옮기자.

Knife 설정으로 서버와 SSL통신을 위해 Key 파일을 생성하였다.

chefadmin.pem, chef-validator.pem, knife.rb 파일을 생성하여 저장한다.

### chefadmin.pem생성

/etc/chef/users/chefadmin.key로 생성된 파일을 복사해서 chefadmin.pem으로 저장한다.

### Knife.rb

current\_dir = File.dirname(\_\_FILE\_\_)

log\_level :info

log\_location STDOUT

node\_name "chefadmin "

client\_key "#{ENV['HOME']}/.chef/chefadmin.pem"

syntax\_check\_cache\_path "#{ENV['HOME']}/.chef/syntaxcache"

validation\_key "#{ENV['HOME']}/.chef/chef-validator.pem"

validation\_client\_name "chef-validator"

chef\_server\_url "https://dev1.im.com/organizations/chef"

cookbook\_path ["#{current\_dir}/../cookbooks"]

cache\_type 'BasicFile'

cache\_options( :path => "#{current\_dir}/checksums" )

* node\_name: This specifies the name that knife will use to connect to your Chef server. This should match your user name.
* current\_dir : $HOME 으로 수정 한다.
* client\_key : 웹 콘솔에서 다운로드 받은 RSA private key 파일
* validation\_key : chef\_server\_url 에 대한 접근을 확인하기 위한 RSA private key 파일
* chef\_server\_url : API를 호출할 chef server url

## knife가 제대로 작동하는지 테스트 한다.

### knife ssl check

SSL 통신 상태를 체크 한다.

### knife ssl fetch

WARNING: Certificates from 10.20.10.240 will be fetched and placed in your trusted\_cert

directory (/home/hadmin/.chef/trusted\_certs).

Knife has no means to verify these are the correct certificates. You should

verify the authenticity of these certificates after downloading.

Adding certificate for dev1.im.com in /home/hadmin/.chef/trusted\_certs/dev1\_im\_com.crt

### $ knife ssl check

Connecting to host dev1.im.com:443

Successfully verified certificates from `dev1.im.com'

### $ knife client list

ERROR: Failed to authenticate to https://dev1.im.com/organizations/chef as chefadmin with key /home/hadmin/.chef/chefadmin.pem

Response: Invalid signature for user or client 'chefadmin'

\* 키파일이 맞지 않는 경우 위와 같은 에러가 발생한다.

\* 웹콘솔에서 chefadmin의 키를 Users>Reset Key > 키를 재 발급하여 재 저장하고 테스트 해보자.

### $ knife user list

chefadmin

### $ knife client list

chef-validator

chefuser

## knife 환경 설정 하는 다른 방법

### knife configure –i 를 실행 하여, 다음과 같이 설정 과정을 진행한다.

\* 아래 상황은 위와 같은 과정의 일부에 문제가 있어 오류가 나는 경우 knife의 명령과 같이 나오게 된다. 12버전 부터는 knife user create를 직접 이용 하라고 되어 있다.

아래 내용은 복잡한 내용이지만 참고로 정리 하였다.

|  |
| --- |
| $ knife configure -i  WARNING: No knife configuration file found  Where should I put the config file? [/home/hadmin/.chef/knife.rb]  Please enter the chef server URL: [https://dev1.im.com:443]  Please enter a name for the new user: [root] hadmin  Please enter the existing admin name: [admin] hadmin  Please enter the location of the existing admin's private key: [/etc/chef-server/admin.pem]  Please enter the validation clientname: [chef-validator]  Please enter the location of the validation key: [/etc/chef-server/chef-validator.pem]  Please enter the path to a chef repository (or leave blank):  Creating initial API user...  Please enter a password for the new user:  WARNING: IF YOU ARE USING CHEF SERVER 12+, PLEASE FOLLOW THE INSTRUCTIONS UNDER knife user create --help.  You only passed a single argument to knife user create.  For backwards compatibility, when only a single argument is passed,  knife user create assumes you want Open Source 11 Server user creation.  knife user create for Open Source 11 Server is being deprecated.  Open Source 11 Server user commands now live under the knife osc\_user namespace.  For backwards compatibility, we will forward this request to knife osc\_user create.  If you are using an Open Source 11 Server, please use that command to avoid this warning.  FATAL: Cannot find sub command for: 'osc\_user configure -i'  Available subcommands: (for details, knife SUB-COMMAND --help)  \*\* BOOTSTRAP COMMANDS \*\*  knife bootstrap FQDN (options)  \*\* CLIENT COMMANDS \*\*  knife client bulk delete REGEX (options)  knife client create CLIENTNAME (options)  knife client delete CLIENT (options)  knife client edit CLIENT (options)  Usage: /usr/bin/knife (options)  knife client key delete CLIENT KEYNAME (options)  knife client key edit CLIENT KEYNAME (options)  knife client key list CLIENT (options)  knife client key show CLIENT KEYNAME (options)  knife client list (options)  knife client reregister CLIENT (options)  knife client show CLIENT (options)  \*\* CONFIGURE COMMANDS \*\*  knife configure (options)  knife configure client DIRECTORY  \*\* COOKBOOK COMMANDS \*\*  knife cookbook bulk delete REGEX (options)  knife cookbook create COOKBOOK (options)  knife cookbook delete COOKBOOK VERSION (options)  knife cookbook download COOKBOOK [VERSION] (options)  knife cookbook list (options)  knife cookbook metadata COOKBOOK (options)  knife cookbook metadata from FILE (options)  knife cookbook show COOKBOOK [VERSION] [PART] [FILENAME] (options)  knife cookbook test [COOKBOOKS...] (options)  knife cookbook upload [COOKBOOKS...] (options)  \*\* COOKBOOK SITE COMMANDS \*\*  knife cookbook site download COOKBOOK [VERSION] (options)  knife cookbook site install COOKBOOK [VERSION] (options)  knife cookbook site list (options)  knife cookbook site search QUERY (options)  knife cookbook site share COOKBOOK [CATEGORY] (options)  knife cookbook site show COOKBOOK [VERSION] (options)  knife cookbook site unshare COOKBOOK  \*\* DATA BAG COMMANDS \*\*  knife data bag create BAG [ITEM] (options)  knife data bag delete BAG [ITEM] (options)  knife data bag edit BAG ITEM (options)  knife data bag from file BAG FILE|FOLDER [FILE|FOLDER..] (options)  knife data bag list (options)  knife data bag show BAG [ITEM] (options)  \*\* ENVIRONMENT COMMANDS \*\*  knife environment compare [ENVIRONMENT..] (options)  knife environment create ENVIRONMENT (options)  knife environment delete ENVIRONMENT (options)  knife environment edit ENVIRONMENT (options)  knife environment from file FILE [FILE..] (options)  knife environment list (options)  knife environment show ENVIRONMENT (options)  \*\* EXEC COMMANDS \*\*  knife exec [SCRIPT] (options)  \*\* HELP COMMANDS \*\*  knife help [list|TOPIC]  \*\* INDEX COMMANDS \*\*  knife index rebuild (options)  \*\* KNIFE COMMANDS \*\*  Usage: /usr/bin/knife (options)  \*\* NODE COMMANDS \*\*  knife node bulk delete REGEX (options)  knife node create NODE (options)  knife node delete NODE (options)  knife node edit NODE (options)  knife node environment set NODE ENVIRONMENT  knife node from file FILE (options)  knife node list (options)  knife node run\_list add [NODE] [ENTRY[,ENTRY]] (options)  knife node run\_list remove [NODE] [ENTRY[,ENTRY]] (options)  knife node run\_list set NODE ENTRIES (options)  knife node show NODE (options)  \*\* OSC COMMANDS \*\*  knife osc\_user create USER (options)  knife osc\_user delete USER (options)  knife osc\_user edit USER (options)  knife osc\_user list (options)  knife osc\_user reregister USER (options)  knife osc\_user show USER (options)  \*\* PATH-BASED COMMANDS \*\*  knife delete [PATTERN1 ... PATTERNn]  knife deps PATTERN1 [PATTERNn]  knife diff PATTERNS  knife download PATTERNS  knife edit [PATTERN1 ... PATTERNn]  knife list [-dfR1p] [PATTERN1 ... PATTERNn]  knife show [PATTERN1 ... PATTERNn]  knife upload PATTERNS  knife xargs [COMMAND]  \*\* RAW COMMANDS \*\*  knife raw REQUEST\_PATH  \*\* RECIPE COMMANDS \*\*  knife recipe list [PATTERN]  \*\* ROLE COMMANDS \*\*  knife role bulk delete REGEX (options)  knife role create ROLE (options)  knife role delete ROLE (options)  knife role edit ROLE (options)  knife role env\_run\_list add [ROLE] [ENVIRONMENT] [ENTRY[,ENTRY]] (options)  knife role env\_run\_list clear [ROLE] [ENVIRONMENT]  knife role env\_run\_list remove [ROLE] [ENVIRONMENT] [ENTRIES]  knife role env\_run\_list replace [ROLE] [ENVIRONMENT] [OLD\_ENTRY] [NEW\_ENTRY]  knife role env\_run\_list set [ROLE] [ENVIRONMENT] [ENTRIES]  knife role from file FILE [FILE..] (options)  knife role list (options)  knife role run\_list add [ROLE] [ENTRY[,ENTRY]] (options)  knife role run\_list clear [ROLE]  knife role run\_list remove [ROLE] [ENTRY]  knife role run\_list replace [ROLE] [OLD\_ENTRY] [NEW\_ENTRY]  knife role run\_list set [ROLE] [ENTRIES]  knife role show ROLE (options)  \*\* SEARCH COMMANDS \*\*  knife search INDEX QUERY (options)  \*\* SERVE COMMANDS \*\*  knife serve (options)  \*\* SSH COMMANDS \*\*  knife ssh QUERY COMMAND (options)  \*\* SSL COMMANDS \*\*  knife ssl check [URL] (options)  knife ssl fetch [URL] (options)  \*\* STATUS COMMANDS \*\*  knife status QUERY (options)  \*\* TAG COMMANDS \*\*  knife tag create NODE TAG ...  knife tag delete NODE TAG ...  knife tag list NODE  \*\* USER COMMANDS \*\*  knife user create USERNAME DISPLAY\_NAME FIRST\_NAME LAST\_NAME EMAIL PASSWORD (options)  knife user delete USER (options)  knife user edit USER (options)  knife user key create USER (options)  knife user key delete USER KEYNAME (options)  knife user key edit USER KEYNAME (options)  knife user key list USER (options)  knife user key show USER KEYNAME (options)  knife user list (options)  knife user reregister USER (options)  knife user show USER (options) |

# Chef Node 초기 설정

## ssh 설정

node에서 서버로 ssh 접근이 가능하도록 설정한다.

Node 초기 설정

Knife bootstrap –u $USRNAME --sudo $FQDN\_CLIENT\_SERVER

# Chef 서버 설치하고 테스트해보기

참조 : http://gyus.me/?p=370

## vagrant + virtualbox로 5대의 멀티 인스턴스 올리기

Chef 서버의 환경을 이해해 보려면 3개이상 가상머신을 설치해서 해보는게 좋다.  
Vagrant로 5대의 가상머신을 올려보자.  
Vagrant를 설치하고 VagrantFile에 아래와 같이 설정한 후

vagrant up을 실행하면 된다.

vagrant 사용법은 아래의 링크를 참고 하자.

Chef의 테스트 환경을 만드는데 매우 적합한 툴 – Vagrant (http://gyus.me/?p=326)