**1. Vagrant?**

 Vagrant는 가상화 기술을 사용자로 하여금 쓰기 쉽고, 편하게 하여 개발 환경을 손쉽게 구축해주는 도구다. Vagrant를 활용하면 다양한 테스트 및 개발에 필요한 환경을 빠르게 구축할 수 있다는 점이 가장 큰 장점이다. 따라서 다양한 개발환경에서 기능 테스트를 수행하거나, 이전에 제작했던 시스템의 환경과 동일한 환경을 재현하는 데 사용될 수 있다.  과거 구축했던 개발 및 운영 환경을 즉시 재현할 수 있다.

* + 개발자가 만든 VM 이미지를 다른 개발자들과 공유할 수 있다.
  + VM 구성 및 배포를 커맨드 몇 번만으로 빠르게 진행할 수 있다.
  + 개발자가 구성한 VM을 서버에서도 그대로 사용할 수 있다.
  + VM 공유 기능을 제공한다.
  + 설치가 매우 쉽다.
  + 소프트웨어 구성 비용이 없다.
  + 유지보수 비용을 최소화할 수 있다.

 그러나 Vigrant는 오픈스택과 같은 Cloud IaaS를 구성하지 않는다. Network, Storage, Compute 가상화를 포함하는 가상화 기술을 제공하지는 않는다. 또한 그럴싸한 Web UI 제공이 안되며, VMware를 기반으로 사용하기 위해서는 유료 플러그-인을 구매해야하는 점이 있다.

**2. Vagrant 사용과 Box에 대한 이해**

 Vagrant 사용에 앞서 선행되어야 할 부분이 일부 있다. 먼저 하이퍼바이저 VirtualBox를 [**다운**](https://www.virtualbox.org/wiki/Downloads)받아서 설치하도록 하자. 그 다음 마찬가지로 Vagrant 공식 페이지에서 제공하는 파일을 OS에 맞게 [**다운**](https://www.vagrantup.com/downloads.html)받아 설치하도록 한다. (필자는 Windows10 환경에서 진행했다.) 이렇게 두 개의 설치만 진행되면 사용을 위한 준비는 끝났다.

 이제 명령프롬프트를 켜서 아래의 두 명령어를 치면 VM이 자동적으로 설치된다. Vagrant 공식 페이지에서 제공하는 문서에서는 Root 디렉토리에 새로운 폴더를 만들어, 그 안에서 아래의 명령어를 수행하는 것을 권장하고 있다. 아래의 화면처럼 나온다면 설치가 정상적으로 진행된 것이다. 아마 대부분 눈치챌 수 있듯이, init 명령어 뒤에 들어가는 인수에 따라 어떤 버전의 OS로 설치할 지 결정한다. 각각의 OS를 담고 있는 이미지를 BOX라고 칭하는 데, 본 포스팅에서는 Ubuntu-12.04 64bit 버전인 precise64로 VM을 생성하고 있다. 아래는 이미지 주소싸이트

https://atlas.hashicorp.com/boxes/search?utf8=%E2%9C%93&sort=&provider=virtualbox&q=precise64

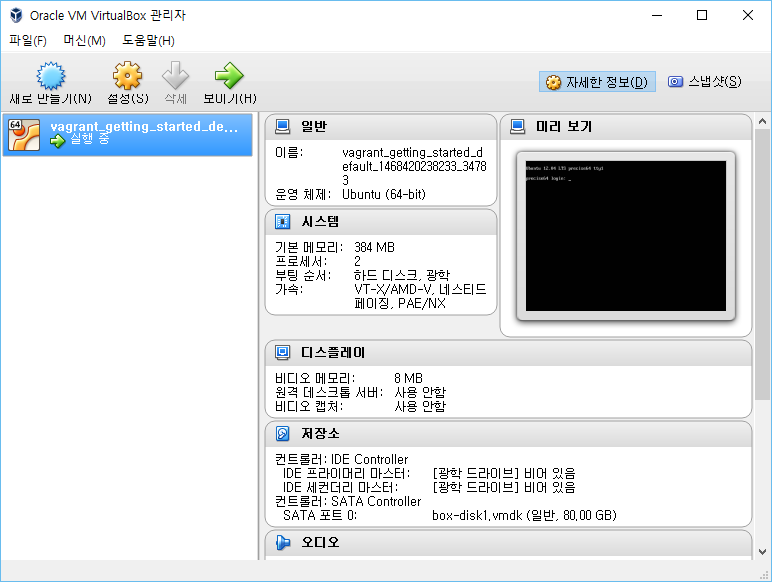
vagrant init hashicorp/precise64

vagrant up

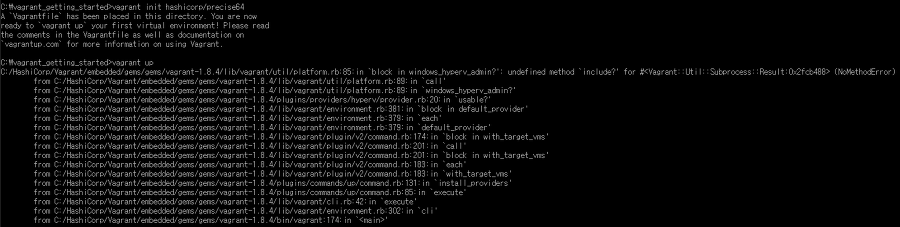


 앞서 말한 것과 마찬가지로 vagrant init 명령어는 뒤에 작성된 URL로부터 VM 세팅을 위한 템플릿을 가져온다. [**https://atlas.hashicorp.com/boxes/search**](https://atlas.hashicorp.com/boxes/search)에 접속해보면, vagrant에서 제공하고 있는 OS별, 구성요소별로 Box 설정을 위한 파일이 제공되고 있다. 이들 중 VM 환경으로 사용을 원하는 Box를 선택하고, init 명령어를 통해 해당 정보를 가져옴으로써 원하는 시스템 환경을 구축할 수 있다. 그리고 vagrant up 명령어를 통해 가져온 설정 정보를 토대로 VM을 생성해준다.

 아래의 화면을 보면 Virtual Box를 설치한 직후 아무것도 없는 상태에서 아래와 같이 하나의 로지컬 머신이 생성된 것을 확인할 수 있다.



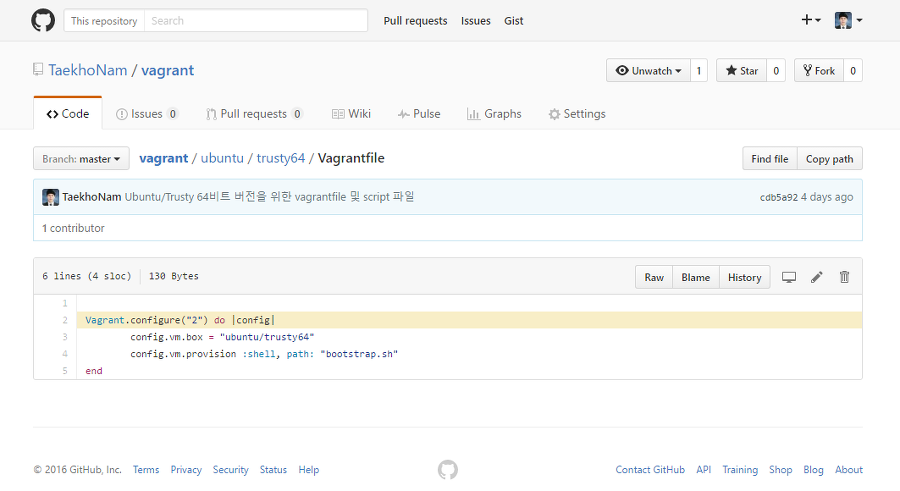
 vagrant init 명령어를 치면, 해당 디렉토리에 'Vagrantfile' 파일이 생성된다. 이전에 Box가 VM 생성을 위한 템플릿이었다면, vagrantfile은 생성될 VM에 대한 세부 설정을 정의한다고 볼 수 있다. VM을 생성할 때, 어떤 box 파일을 사용할 것인지, VM에 할당할 CPU, Memory, Network 등의 configuration을 vagrantfile에서 정의 및 수정할 수 있다.



 'vagrant up' 명령어를 실행할 때 위와 같은 오류가 발생한다면 이는 Virtual Box 버전에 따른 오류다. 필자도 아래와 같은 오류를 해결하는데 시간을 꽤 많이 할애 했는데, 설정을 잘못 했다거나(사실 설정이라 할 것도 없지만..), 설치 과정에서 문제가 있었던 것은 아니었다. 단지 갱신된 Virtual Box 버전에 따라 동작하지 않았던 것이다. Vagrant 동작을 확인한 버전은 [**Virtualbox 5.0.24**](http://download.virtualbox.org/virtualbox/5.0.24/)이므로 해당 버전으로 설치할 것을 권고하는 바이다.

**3. VM Provisioning**

 Vagrant를 사용하여 VM을 생성한다고 해도, 완벽한 Provisioning이 진행됐다고 보긴 어렵다. 각 VM에서 개발환경에 필요한 소프트웨어 설치가 수동으로 필요하다면, 이는 절반뿐인 Provisioning이다. 따라서 웹 서버, 미들웨어, DB등을 설치하고, Configuration 하기위한 방법이 필요하다. 물론 VM 생성을 위한 이미지 자체에 이들을 포함시켜서 배포하는 경우도 있지만, 이 경우 한 종류의 개발 환경만 배포되므로 유연한 사용은 어렵다.



 Vagrant는 이런 불편함을 해소할 수 있는 Provisioning 기능을 제공한다. VM 생성 후, vagrantfile에 기술된 provisioning script를 수행함으로써 필요한 소프트웨어를 자동으로 설치하고, configuration해준다. 위 사진처럼 Vagrantfile 작성 시, VM 시스템 환경을 구축하자마자 bootstrap.sh에 작성된 스크립트 언어에 맞춰 필요한 소프트웨어를 설치하고 설정해준다. 한 가지 주의할 점은 vagrantfile의 provisioning 부분에 기술된 명령어는 vagrant up, reload, provision 세 개의 멸령어가 실행될 때마다 매번 실행된다. 따라서 스크립트 내에 해당 소프트웨어가 설치되었는지 확인 후에, 설치가 안된 소프트웨어에 대해서만 설치를 진행하다록 스크립트를 작성하는 것이 좋다.

# **Box 개념 이해하기**

앞에서 vagrant init 명령을 실행할때, preceise32.box라는 파일을 지정하였다. 이 box 파일은 VM을 만들기 위한 기본 OS 이미지를 포함한 VM 설정(CPU,메모리 사이즈등)에 대한 기본 템플릿이다. (사이즈가 보통 수백 메가가 나간다.)

<http://www.vagrantbox.es/> 에 보면 공개된 box 파일들이 있다. Ubuntu, Debian 등 다양한 Linux OS 버전의 VM 들에 대한 box 파일들이 있다.

**Vagrant file**

Vagrant init을 하면, 해당 디렉토리에 “Vagrantfile” 이라는 이름으로 생성되는 파일인데, Box가 VM 생성을 위한 기본 템플릿이라면, Vagrant file은 생성될 VM에 대한 세부 설정을 정의한다. VM을 생성할때, 어떤 box 파일을 사용할 것인지, VM에 대한 하드웨어 설정 예를 들어 CPU,메모리 사이즈,네트워크, 네트워크 포트포워딩 설정등을 여기서 재정의 할 수 있다.

아래는 Oracle Virtual Box실행시 preceise32 box 이미지를

<http://files.vagrantup.com/precise32.box> 에서 읽어와서, CPU 2개, 512M를 가진 “Terry\_vargrant0”이라는 VM을 생성하는 Vagrantfile이다. 아래와 같이 파일을 생성한후에, vagrant up 명령을 수행시키면 설정한 정보 대로 VM이 생성된다.

|  |
| --- |
| VAGRANTFILE\_API\_VERSION = "2"    Vagrant.configure(VAGRANTFILE\_API\_VERSION) do |config|    config.vm.box = "precise32"    config.vm.box\_url = "http://files.vagrantup.com/precise32.box"    # config.vm.network :forwarded\_port, guest: 80, host: 8080    # config.vm.network :private\_network, ip: "192.168.33.10"    # config.vm.network :public\_network    # config.ssh.forward\_agent = true    config.vm.provider "virtualbox" do |vm|          vm.customize [                 "modifyvm",:id,                 "--memory","512",                 "--name","Terry\_vagrant0",                 "--cpus","2",                         ]    end  end |

**Vagrant + Provisioning**

Vagrant를 이용하면, VM을 쉽게 만들 수 있다. 그런데 개발환경을 구축하자면, OS가 인스톨된 VM 뿐만 아니라, 그위에 웹서버,DB등 미들웨어들을 설치해야 하고, 그리고 거기에 맞는 Configuration을 해야 한다. 물론 미리 VM 이미지에 웹서버등을 설치해놓고, 필요에 따라서 vagrant를 이용해서 해당 VM들을 설치해서 사용해야 하지만 그 경우에는 설정마다 매번 다른 VM이미지를 만들어놔야 하기 때문에 번거롭다. 만약에 OS 만 설치된 VM에다가, 설정에 따라서 소프트웨어와 설정을 하는 부분을 분리한다면?

이런 접근을 지원하는 기능이 Vagrant에 provisioning이라는 기능이 있다. VM을 기동한 후에, vagrantfile에 정의된 provisioning script를 수행해준다. 다음 예제를 보자. 다음 예제는 VM이 기동된 후에, apt-get 명령을 이용해서 apache2 (웹서버)를 자동으로 설치하는 설정이다.

|  |
| --- |
| VAGRANTFILE\_API\_VERSION = "2"    Vagrant.configure(VAGRANTFILE\_API\_VERSION) do |config|    config.vm.box = "precise32"    config.vm.box\_url = "http://files.vagrantup.com/precise32.box"  config.vm.provision :shell, :inline => "sudo apt-get install -y apache2"    end |

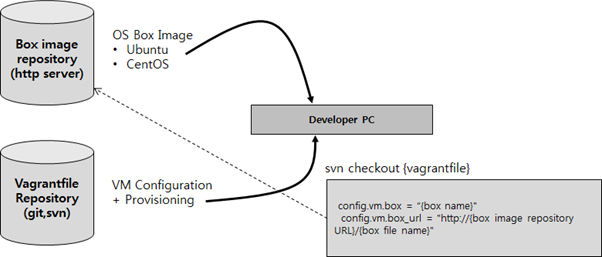
위의 예제는 VM이 기동될때 마다 shell 명령어를 수행하도록 한것인데, 명령어말고도 shell스크립트를 수행하게 할 수 도 있고, puppet이나 chef와 같은 configuration management 도구를 이용해서, 제품을 설치하게 할 수 도 있다.

한 가지 주의할점은 Vagrantfile의 provision 부분에 정의된 명령어는 vagrant up, reload, provision 3개의 명령어가 실행될때 마다 매번 실행된다. up에서도 매번 실행되기 때문에, 스크립트내에, 해당 소프트웨어가 미리 설치되었는지 확인한 후에, 설치가 안되어 있을 경우에만 설치하도록 스크립트를 짜는 것이 좋다.

Provisioning에 대한 자세한 방법은 <http://docs.vagrantup.com/v2/provisioning/index.html> 를 참고하면 된다.

**Vagrant를 이용한 개발 환경 구축**

그러면 Vagrant를 이용해서 개발환경을 어떻게 구축할 수 있는지 살펴보도록 하자



크게 그림과 같이 2개의 repository가 필요하다. Box image repository에는 기본 이미지가 인스톨된 box image들을 저장해놓는다.

그리고 svn이나 git와 같은 VCS 툴에 vagrantfile을 저장해놓는다. (아니면 간단하게 웹서버에 저장해놔도 된다.) 이 Vagrantfile에는 box 파일들을 저장해놓은 repository를 pointing 하도록 하고, 필요에 따라서

1.  Ubuntu + Apache

2.  Ubuntu + MySQL

3.  Ubuntu + Tomcat

와 같이 다양한 설정을 만들어 놓고, 필요에 따라서 Vagrantfile이 받은 후에, 간단하게 “vagrant up” 명령어만 수행하면 간단하게 개발환경에 필요한 VM을 만들어낼 수 있다.

지금까지 간략하게 Vagrant에 개념과 사용법에 대해서 알아보았다.Vagrant는 그외에도, Vagrant는 단일 VM 뿐만 아니라 multi vm을 단일 vagrantfile에서 설정이 가능하고, Oracle의 Virtual Box뿐만 아니라,VMWare 및 Amazon EC2 클라우드 까지 지원한다. 간단하게는 개발환경에서 부터,응용하면, QA,스테이징,운영환경 배포용으로도 활용할 수 있다.

자세한 내용들은 <http://docs.vagrantup.com/> 를 참고하기 바란다.

**Box 개념 이해하기**

앞에서 vagrant init 명령을 실행할때, preceise32.box라는 파일을 지정하였다. 이 box 파일은 VM을 만들기 위한 기본 OS 이미지를 포함한 VM 설정(CPU,메모리 사이즈등)에 대한 기본 템플릿이다. (사이즈가 보통 수백 메가가 나간다.)

<http://www.vagrantbox.es/> 에 보면 공개된 box 파일들이 있다. Ubuntu, Debian 등 다양한 Linux OS 버전의 VM 들에 대한 box 파일들이 있다.

**Vagrant file**

Vagrant init을 하면, 해당 디렉토리에 “Vagrantfile” 이라는 이름으로 생성되는 파일인데, Box가 VM 생성을 위한 기본 템플릿이라면, Vagrant file은 생성될 VM에 대한 세부 설정을 정의한다. VM을 생성할때, 어떤 box 파일을 사용할 것인지, VM에 대한 하드웨어 설정 예를 들어 CPU,메모리 사이즈,네트워크, 네트워크 포트포워딩 설정등을 여기서 재정의 할 수 있다.

아래는 Oracle Virtual Box실행시 preceise32 box 이미지를

<http://files.vagrantup.com/precise32.box> 에서 읽어와서, CPU 2개, 512M를 가진 “Terry\_vargrant0”이라는 VM을 생성하는 Vagrantfile이다. 아래와 같이 파일을 생성한후에, vagrant up 명령을 수행시키면 설정한 정보 대로 VM이 생성된다.

|  |
| --- |
| VAGRANTFILE\_API\_VERSION = "2"    Vagrant.configure(VAGRANTFILE\_API\_VERSION) do |config|    config.vm.box = "precise32"    config.vm.box\_url = "http://files.vagrantup.com/precise32.box"    # config.vm.network :forwarded\_port, guest: 80, host: 8080    # config.vm.network :private\_network, ip: "192.168.33.10"    # config.vm.network :public\_network    # config.ssh.forward\_agent = true    config.vm.provider "virtualbox" do |vm|          vm.customize [                 "modifyvm",:id,                 "--memory","512",                 "--name","Terry\_vagrant0",                 "--cpus","2",                         ]    end  end |

**Vagrant + Provisioning**

Vagrant를 이용하면, VM을 쉽게 만들 수 있다. 그런데 개발환경을 구축하자면, OS가 인스톨된 VM 뿐만 아니라, 그위에 웹서버,DB등 미들웨어들을 설치해야 하고, 그리고 거기에 맞는 Configuration을 해야 한다. 물론 미리 VM 이미지에 웹서버등을 설치해놓고, 필요에 따라서 vagrant를 이용해서 해당 VM들을 설치해서 사용해야 하지만 그 경우에는 설정마다 매번 다른 VM이미지를 만들어놔야 하기 때문에 번거롭다. 만약에 OS 만 설치된 VM에다가, 설정에 따라서 소프트웨어와 설정을 하는 부분을 분리한다면?

이런 접근을 지원하는 기능이 Vagrant에 provisioning이라는 기능이 있다. VM을 기동한 후에, vagrantfile에 정의된 provisioning script를 수행해준다. 다음 예제를 보자. 다음 예제는 VM이 기동된 후에, apt-get 명령을 이용해서 apache2 (웹서버)를 자동으로 설치하는 설정이다.

|  |
| --- |
| VAGRANTFILE\_API\_VERSION = "2"    Vagrant.configure(VAGRANTFILE\_API\_VERSION) do |config|    config.vm.box = "precise32"    config.vm.box\_url = "http://files.vagrantup.com/precise32.box"  config.vm.provision :shell, :inline => "sudo apt-get install -y apache2"    end |

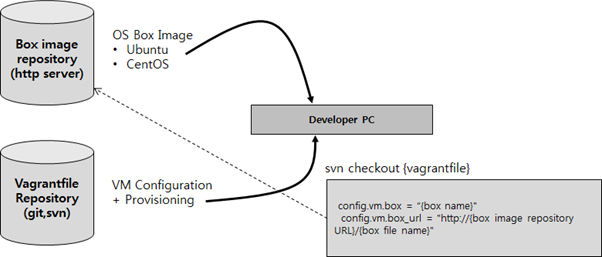
위의 예제는 VM이 기동될때 마다 shell 명령어를 수행하도록 한것인데, 명령어말고도 shell스크립트를 수행하게 할 수 도 있고, puppet이나 chef와 같은 configuration management 도구를 이용해서, 제품을 설치하게 할 수 도 있다.

한 가지 주의할점은 Vagrantfile의 provision 부분에 정의된 명령어는 vagrant up, reload, provision 3개의 명령어가 실행될때 마다 매번 실행된다. up에서도 매번 실행되기 때문에, 스크립트내에, 해당 소프트웨어가 미리 설치되었는지 확인한 후에, 설치가 안되어 있을 경우에만 설치하도록 스크립트를 짜는 것이 좋다.

Provisioning에 대한 자세한 방법은 <http://docs.vagrantup.com/v2/provisioning/index.html> 를 참고하면 된다.

**Vagrant를 이용한 개발 환경 구축**

그러면 Vagrant를 이용해서 개발환경을 어떻게 구축할 수 있는지 살펴보도록 하자



크게 그림과 같이 2개의 repository가 필요하다. Box image repository에는 기본 이미지가 인스톨된 box image들을 저장해놓는다.

그리고 svn이나 git와 같은 VCS 툴에 vagrantfile을 저장해놓는다. (아니면 간단하게 웹서버에 저장해놔도 된다.) 이 Vagrantfile에는 box 파일들을 저장해놓은 repository를 pointing 하도록 하고, 필요에 따라서

1.  Ubuntu + Apache

2.  Ubuntu + MySQL

3.  Ubuntu + Tomcat

와 같이 다양한 설정을 만들어 놓고, 필요에 따라서 Vagrantfile이 받은 후에, 간단하게 “vagrant up” 명령어만 수행하면 간단하게 개발환경에 필요한 VM을 만들어낼 수 있다.

지금까지 간략하게 Vagrant에 개념과 사용법에 대해서 알아보았다.Vagrant는 그외에도, Vagrant는 단일 VM 뿐만 아니라 multi vm을 단일 vagrantfile에서 설정이 가능하고, Oracle의 Virtual Box뿐만 아니라,VMWare 및 Amazon EC2 클라우드 까지 지원한다. 간단하게는 개발환경에서 부터,응용하면, QA,스테이징,운영환경 배포용으로도 활용할 수 있다.

자세한 내용들은 <http://docs.vagrantup.com/> 를 참고하기 바란다.

### **vagrant 명령어** UP AND SSH

*vagrant up*

* 현재 Vagrantfile 환경을 부팅한다.

*vagrant ssh*

* /vagrant 폴더와 Vagrantfile 이있는 폴더가 sync 가 된다. 파일을 삭제하면 Host 에서도 적용이된다.

*vagrant destroy*

* up 에 적용됬던 내용들을 모두 지운다. **box 를 지우는건 아니다.**

Synced Folders

* 터미널을 통해 소스를 수정하는건 매우 귀찮은일 따라서 Vagrant 는 Host 컴퓨터와 폴더를 공유한다.
* 기본적으로 호스트의 Vagrantfile 이 있는 폴더와 vagrant 의 **/vagrant** 폴더를 공유한다.

Provision

* 게스트 머신을 반복적으로 생성하고 사용하기 위해 프로비져닝으로 만들어서 사용한다. 내부적으로 지원하는 자동 프로비져닝을 통해 자동적으로 소프트웨어를(nginx mysql 등등) 을 설치한다.

1 *#!/usr/bin/env bash*

2

3 apt-get update

4 apt-get install -y apache2

5 **if** ! **[** -L /var/www **]**; **then**

6 rm -rf /var/www

7 ln -fs /vagrant /var/www

8 **fi**

1 Vagrant**.**configure("2") **do** **|**config**|**

2 config**.**vm**.**box **=** "hashicorp/precise32"

3 config**.**vm**.**provision :shell, path: "bootstrap.sh"

4 **end**

* bootstrap.sh 스크립트를 만들어주고 Vagrantfile 에서 provision 으로 shell 을 지저앟고 bootstrap.sh 를 설치하게 해준다.
* provision 은 **vagrant up** 을 실행할때 만들어 지므로 이미 실행중일때는 **vagrant reload –provision** 으로 reload 해준다.

Networking

* Port Forwarding : Guest machine 에 포트와 Host macine 의 포트를 연결해준다.

1 Vagrant**.**configure("2") **do** **|**config**|**

2 config**.**vm**.**box **=** "hashicorp/precise32"

3 config**.**vm**.**provision :shell, path: "bootstrap.sh"

4 *# 호스트의 4567 를 게스트 80으로 연결*

5 *# 브라우져에서 127.0.0.1:4567 연결하면*

6 *# 호스트의 apache 80번 포트로 연결된다.*

7 config**.**vm**.**network :forwarded\_port, host: 4567, guest: 80

8 **end**

Share

* 내가 만든 Vagrantfile 을 공유한다.

*vagrant login vagrant share*

Teardown

*vagrant suspend*

* 현재 실행 상태를 저장하고 종료한다. 다시 vagrant up 하면 10초면 다시 실행 시킬수있다. 디스크와 램을 사용하고 있는 상태로 종료된다.

*vagrant halting*

* 컴퓨터를 종료하는것과 같다. 디스크에는 남겨두고 램에서는 삭제한다. vagrant up 하면 다시 램에 올리고 실행한다.

*vagrant destroy*

* 완전히 디스크에서 지워버린다. vagrant up 하면 다시 설치해서 실행하므로 오래걸린다.

Providers

*vagrant up –provider=vmware\_fusion vagrant up –provider=aws*

* provider 는 기본으로는 virutalbox 지만 바꿀수도 있다.

CLI

* vagrant box
  + vagrant box add <address> : 박스 추가
  + vagrant box list : 설치된 박스 리스트
  + vagrant box outdated : 최신 업데이트 된 박스들 보여주기
  + vagrant remove <address> : 박스 삭제
  + vagrant repackage :
  + vagrant update :
* vagrant destroy : 완전히 디스크에서 삭제
* vagrant global-status : 현재 켜져 있는 Vagrant environments 를 보여준다. vagrant halt 등으로 꺼줘야 한다.
* vagrant halt : vagrant 꺼준다
* vagrant package : 현재 실행중인 Virtualbox 를 재사용 가능한 box로 만든다.
* vagrant plugin
  + vagrant plugin install
  + vagrant plugin license
  + vagrant plugin list
  + vagrant plugin uninstall
  + vagrant plugin update
* vagrant ssh : ssh 접속
* vagrant status