

Vagrant

Creación y configuración automatizada de máquinas virtuales



¿Qué permite?

- Desplegar el SO que necesitemos para nuestra aplicación de forma rápida y cómoda.
- Nos ayuda a simular nuestro entorno final de ejecución
- Reproducir configuración exacta de MySQL, Linux, Gems, Apache, Passenger...

¿Qué permite?

- Guardar 'boxes' para empezar justo con lo que necesitamos rápidamente en cualquier sitio.
- Posibilidad de publicarlas fácilmente
- 'Boxes' oficiales: Ubuntu Lucid y Precise en 32 y 64 bits
- Contribuidas: Un amplio abanico de distribuciones

¿Qué necesito?

- Una máquina con *nix, preferiblemente Linux o MacOS con Ruby(1.9.3) instalado.
- Aprender unos pocos comandos.
- Mirar en la documentación las opciones del Vagrantfile.

Nota para usuarios de Mac

- Es primordial tener GNU GCC instalado. El compilador de C/C++ del Xcode actual (llvm) *no* sirve y dará error al instalar vagrant y gemas varias.
 - URL: <https://github.com/kennethreitz/osx-gcc-installer>
- Requiere Ruby 1.9.3

Comenzando

- `gem install vagrant`
- Importar boxes a nuestra máquina local
 - `vagrant box add precise64`
<http://files.vagrantup.com/precise64.box>

[vagrantbox.es]

Comenzando: primera VM

- Crear un directorio
- `Vagrant init boxdepartida`
- Editar el Vagrantfile de acuerdo a nuestras necesidades
- `Vagrant up`, en la misma carpeta del Vagrantfile

Personalizando nuestra máquina

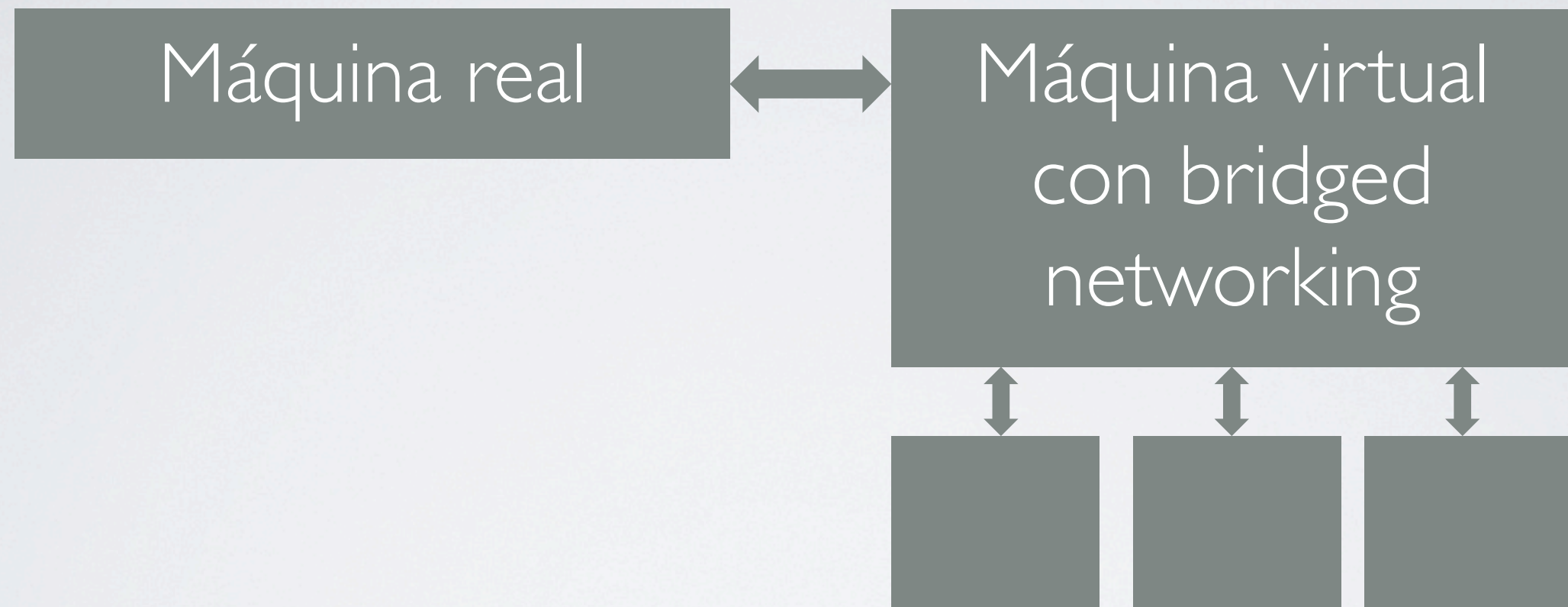
```
Vagrant::Config.run do |config|
  config.vm.box = "base"
  config.vm.box_url = "http://domain.com/path/to/above.box"
  # config.vm.boot_mode = :gui
  config.vm.network :hostonly, "33.33.33.10"
  #config.vm.network :bridged
  config.vm.forward_port 80, 8080
  #identificador, path en el guest, path en el host
  config.vm.share_folder "v-data", "/vagrant_data", "../data"
end
```

`config.vm.network :bridged, :bridge => 'eth1', :mac => '00:11:22:33:44:55'`

Carpetas compartidas

- Interesantes a la hora de automatizar nuestro flujo de trabajo, ya sea con una o varias máquinas
- A través de ellas podemos introducir en las máquinas virtuales los ficheros de la aplicación que estemos probando, esquemas de base de datos...
- Conviene que su contenido provenga de un sistema de control de versiones.

Un ejemplo con varias máquinas



Máquinas con host
only (NAT)

Vagrantfile con varias VM

```
Vagrant::Config.run do |config|
  config.vm.define :app do |app_config|
    app_config.vm.customize ["modifyvm", :id, "--name", "app", "--memory", "512"]
    app_config.vm.box = "precise64"
    app_config.vm.host_name = "app"
    app_config.vm.forward_port 22,2222, :auto => true # auto trata de buscar otro puerto si encuentra colisiones
    app_config.vm.forward_port 80,4567
    app_config.vm.network :hostonly "33.33.33.37"
    #app_config.vm.network :bridged
    # Cuando usamos bridged networking, vagrant crea también automáticamente un interfaz host-only
    # para poder conectar con la máquina física cuando hacemos un vagrant ssh
  end
  config.vm.define :dbmaster do |db_master_config|
    db_master_config.vm.customize ["modifyvm", :id, "--name", "dbmaster", "--memory", "256"]
    db_master_config.vm.box = "precise64"
    db_master_config.vm.host_name = "dbmaster"
    db_master_config.vm.forward_port 22,2222, :auto => true
    db_master_config.vm.network :hostonly "33.33.33.38"
  end
  config.vm.define :dbslave do |db_slave_config|
    db_slave_config.vm.customize ["modifyvm", :id, "--name", "dbslave", "--memory", "256"]
    db_slave_config.vm.box = "precise64"
    db_slave_config.vm.host_name = "dbslave"
    db_slave_config.vm.forward_port 22,2222, :auto => true
    db_slave_config.vm.network :hostonly "33.33.33.39"
  end
end
end
```


Necesito más: veewee

- Si no nos sirven las boxes hechas, podemos hacer las nuestras con VeeWee
- Se instala como gema
- Plantillas para Solaris, Windows/Windows Server, NetBSD, Linux...
- Pocos comandos. Fácil de personalizar.

Ejemplo: generando una box CentOS

- Gem install veewee
- vagrant basebox templates
- Creamos una carpeta con una subcarpeta llamada 'iso'. Ponemos ahí la iso correspondiente.
- Suponiendo que hayamos descargado CentOS-6.3-i386-minimal.iso, `vagrant basebox define 'centos63' 'CentOS-6.3-i386-minimal'`
- Editar definitions/centos63/definition.rb

Ejemplo: generando una box CentOS

- `vagrant basebox build centos63`
- `vagrant basebox validate centos63`
- `vagrant basebox export centos63` // podemos ponerla en un servidor web, por ejemplo
- `vagrant box add centos63 centos63.box` // añadirla a nuestras boxes vagrant locales

Veewee, ventajas adicionales

- También es una herramienta muy interesante para probar archivos de preseed de Debian o Ubuntu, o Kickstart para instalaciones basadas en RedHat/CentOS.

Preseed y Kickstart

- <http://wiki.debian.org/DebianInstaller/Preseed>
- http://fedoraproject.org/wiki/Anaconda/Kickstart#How_Do_You_Perform_a_Kickstart_Installation

Creando una 'base box' manualmente

- En la web de Vagrant hay una sección dedicada a comentar los convenios que se siguen en las base boxes.
- http://vagrantup.com/v1/docs/base_boxes.html

Provisionamiento, proveedores

- Puppet (server, local)
- Chef (server, local)
- Shell (mi propio shell script)
- Otro (escrito por mí en Ruby)

Basado en scripts

- Desventaja: se ejecuta cada vez que encendamos la máquina, luego hay que controlar lo que ocurre a partir de la segunda vez que se ejecuta el script manualmente.
- Ventaja: La opción más inmediata y sencilla


Puppet

- Basa el provisionamiento en una serie de hechos que tienen que darse o 'facts'. Se apoya en una herramienta llamada 'facter'.
- Podemos estructurar las reglas que queremos que nuestro sistema cumpla en clases y módulos para reaprovechar código.

Puppet

- ¿Qué gestiona?
 - Paquetes
 - Servicios
 - Tareas del cron
 - Bases de datos
 - Otros (Iptables, Nagios, Vlans, Keys, SSH...)

Puppet Labs: Puppet Forge

[Documentation](#) [Support](#) [Contact Us](#) [Download](#)

[Puppet](#) [Services](#) [Resources](#) [Community](#) [Company](#)

Puppet Forge

PuppetForge 0.7.0

User: Puppet Labs (puppetlabs)

66 modules found.

puppetlabs/activemq
Version 0.1.6 · June 21, 2011 · activemq, amqp, java, mcollective, middleware, stdlib, stomp

puppetlabs/apache
Version 0.4.0 · August 24, 2012 · apache, virtualhost, web

puppetlabs/appdirector
Version 0.0.1 · August 24, 2012 · vmware

puppetlabs/apt
Version 1.0.1 · October 30, 2012 · apt

puppetlabs/bacula
Version 0.0.2 · December 15, 2011 · backup, bacula

puppetlabs/boundary
Version 1.0.3 · July 27, 2012 · boundary, bprobe, network, probe

puppetlabs/bprobe
Version 1.0.2 · May 10, 2012 · boundary, bprobe, network, probe

puppetlabs/cloud_provisioner
Version 1.0.5 · July 19, 2012 · amazon, aws, cloud, cloud-provisioner, ec2, provisioner, provisioning, puppet, puppetlabs

puppetlabs/cloudformation
Version 0.0.2 · April 10, 2012 · amazon, cfn, cloudformation, ec2, enterprise, pe, puppet

puppetlabs/collectd
Version 0.0.1 · May 20, 2010 · collectd, RRD, statistics

[Sign in](#)
[Register](#)

Find Modules

[Go](#)

Modules

[Add a module](#)

[All Modules](#)
[Web Servers](#)
[Applications](#)
[Package Management](#)
[Operating Systems](#)
[Programming Languages](#)
[Networking](#)
[Utilities](#)
[Monitoring and Trending](#)
[Security](#)
[Virtualization](#)

Popular Tags

[ubuntu](#) (104 modules)
[debian](#) (82 modules)
[CentOS](#) (68 modules)
[rhel](#) (50 modules)
[security](#) (45 modules)
[networking](#) (44 modules)
[applications](#) (40 modules)
[monitoring](#) (38 modules)

Empezando con Puppet

- Usaremos la aproximación basada en ficheros
- Creamos un directorio de ficheros 'manifest'
- Especificamos en el Vagrantfile qué manifiesto queremos ejecutar

Ficheros de configuración

/vagrantproject

Vagrantfile

modules

manifests

module I

nodes.pp

Site.pp

files

manifests

Empezando con Puppet

- En el Vagrantfile:

```
app_config.vm.provision :puppet do |puppet|  
  puppet.manifests_path = "manifests"  
  puppet.manifest_file = "nodes.pp"  
  puppet.module_path = "modules"  
end
```


En site.pp

```
package {  
  "apache2":  
    ensure => present  
}
```

Ejecutando puppet

- Todos los scripts de provisionamiento se ejecutan al encender las máquinas.
- Si queremos ejecutarlos manualmente, podemos ejecutar `vagrant provision`.
- Es importante que especifiquemos un nombre de host con dominio(FQDN) en el Vagrantfile (Si no, tendremos problemas al ejecutar facter)