

Computación en la Nube
Aprovisionamiento con Vagrant
Oscar H. Mondragon

Definición

“En el ámbito de computación en la nube el aprovisionamiento es la acción de contratar, abastecer o asignar recursos computacionales durante un periodo de tiempo determinado. De igual manera estos recursos pueden ser liberados de acuerdo a la necesidad del usuario”

Fuente: https://estrategia.gobiernoonlinea.gov.co/623/articles-75246_recurso_2.pdf

Aprovisionamiento con Vagrant

- Provisioners en Vagrant permiten instalar software automáticamente, alterar configuraciones y más en la máquina como parte del proceso de `vagrant up`
- Usar `vagrant ssh` e instalar software manualmente funciona pero no es lo mas eficiente
- Utilizar los sistemas de aprovisionamiento integrados en Vagrant, automatiza el proceso para que sea repetible
- No requiere interacción humana, por lo tanto puede sin problema hacer `vagrant destroy` y levantar rápidamente su ambiente con `vagrant up`

Quando se hace aprovisionamiento?

El aprovisionamiento ocurre en los siguientes momentos:

- Cuando se ejecuta `vagrant up` la primera vez que se levanta el ambiente
 - Si ejecuta `vagrant up` después de la primera vez, el aprovisionamiento no se realiza, por lo tanto es necesario usar la bandera `--provision`
- Cada vez que se ejecuta `vagrant provision`
- Cuando se ejecuta `vagrant reload --provision`

NOTA: También puede levantar el ambiente y decirle explícitamente que no realice aprovisionamiento mediante la bandera `--no-provision`

Aprovisionadores en Vagrant

- File
- Shell
- Ansible
- Puppet
- Chef
- Docker
- Podman
- Salt

File Provisioner

- Permite cargar un archivo o directorio desde la máquina host a la máquina invitada.
- Ejemplo:

```
Vagrant.configure("2") do |config|
```

```
# ... other configuration
```

```
config.vm.provision "file", source: "~/path/to/host/folder",  
destination: "$HOME/remote/newfolder"
```

```
end
```

File Provisioner

- Las cargas de archivos con File Provisioner se realizan como usuario de SSH o PowerShell.
- Esto es importante ya que estos usuarios generalmente no tienen privilegios elevados por sí mismos.
- Si desea cargar archivos en ubicaciones que requieren privilegios elevados, se recomienda que los cargue en ubicaciones temporales y luego use el aprovisionador `shell` para moverlos a su lugar.

Shell Provisioner

- Permite cargar y ejecutar un script dentro de la máquina invitada.
- Permite aprovisionar haciendo uso de las opciones
 - **inline**: configuración dentro del mismo Vagrantfile
 - **path**: configuración haciendo uso de un archivo Shell externo, el cual se carga y ejecuta

Inline Script

- Un inline script es un script que se le da a Vagrant directamente dentro del Vagrantfile
- Ejemplo:

```
Vagrant.configure("2") do |config|  
    config.vm.provision "shell", inline: "echo Hello, World"  
end
```

- Esto hace que `echo Hello, World` se ejecute dentro de la máquina invitada cuando se ejecute el aprovisionamiento

Inline Script con Ruby heredoc

- Usando *ruby heredoc* se puede agregar un bloque que inicia por ejemplo con `<<-SCRIPT` y termina con `SCRIPT`
- Ejemplo:

```
$script = <<-SCRIPT
echo I am provisioning...
date > /etc/vagrant_provisioned_at
SCRIPT
Vagrant.configure("2") do |config|
    config.vm.provision "shell", inline: $script
end
```

- Mas documentación en https://ruby-doc.org/core-2.5.0/doc/syntax/literals_rdoc.html#label-Here+Documents

Shell Usando Script Externos

- El proveedor de shell también puede tomar una opción que especifica una ruta a un script de shell en la máquina host. Vagrant luego cargará este script en el invitado y lo ejecutará. Por ejemplo:

```
Vagrant.configure("2") do |config|  
    config.vm.provision "shell", path: "script.sh"  
end
```

- Las rutas relativas, como las anteriores, se expanden en relación con la ubicación del Vagrantfile de su proyecto. También se pueden utilizar rutas absolutas

Ejemplo Script Shell

```
#!/bin/bash

echo "configurando el resolv.conf con cat"
cat <<TEST> /etc/resolv.conf
nameserver 8.8.8.8
TEST

echo "instalando un servidor vsftpd"
sudo apt-get install vsftpd -y

echo "Modificando vsftpd.conf con sed"
sed -i 's/#write_enable=YES/write_enable=YES/g' /etc/vsftpd.conf

echo "configurando ip forwarding con echo"
sudo echo "net.ipv4.ip_forward = 1" >> /etc/sysctl.conf
```

Shell Usando Script Externos (url)

- Si usa un script remoto como parte de su proceso de aprovisionamiento, también puede pasar su URL como argumento de ruta:

```
Vagrant.configure("2") do |config|  
    config.vm.provision "shell", path:  
    "https://example.com/provisioner.sh"  
end
```

Argumentos en los Scripts

- Puede parametrizar sus scripts así como cualquier script de shell normal. Estos argumentos se pueden especificar en el proveedor de Shell. Ejemplo

```
Vagrant.configure("2") do |config|  
  config.vm.provision "shell" do |s|  
    s.inline = "echo $1"  
    s.args = ["hello, world!"]  
  end  
end
```

Ansible Provisioner

- Ansible es un poderoso lenguaje de automatización, despliegue y orquestación de código abierto.
- Instalar Ansible en Host Ubuntu

```
sudo apt-get install ansible
```

- Instalar Ansible en Host Windows: **Ansible no corre directamente en un host Windows** pero puede correr bajo el Windows Subsystem for Linux (WSL) (Ver referencia al final)

Ansible Provisioner

- Playbook File: contiene los pasos que se deben correr en el guest. Ejemplo (playbook.yml):

- hosts: all

become: true

tasks:

- name: actualizar apt cache

apt: update_cache=yes

- name: instalar apache

apt: name=apache2 state=present

- name: ejecutar un script de prueba

script: prueba.sh

Playbook

En el playbook de la diapositiva anterior estamos realizando lo siguiente:

- Ejecute este script en todos los boxes conocidos (`hosts: all`)
- Estas tareas requieren sudo (`become: true`)
- Las tareas a ejecutar serán:
 - actualice apt cache (`apt: update_cache=yes`)
 - instale apache (`apt: name=apache2 state=present`)
 - ejecute un bash script (`script: prueba.sh`)

Playbook

El contenido de del bash script puede ser, por ejemplo algo tan simple como iniciar el servicio de apache y ejecutar un comando wget para comprobar el servicio y copiar el resultado en un archivo de texto en el directorio sincronizado /vagrant/

```
#!/bin/bash
```

```
systemctl start apache2
```

```
curl 127.0.0.1 > /vagrant/resultado.txt
```

Corriendo Ansible

Especificar el ansible playbook

```
Vagrant.configure("2") do |config|  
  config.vm.define :vm_ansible do |vm_ansible|  
    vm_ansible.vm.box = "ubuntu/trusty64"  
    vm_ansible.vm.provision :ansible do |ansible|  
      ansible.playbook = "playbook.yml"  
    end  
  end  
end
```

Correr Ansible instalandolo en el guest

De esta forma no tendremos que instalarlo en el anfitrión windows!

```
# -*- mode: ruby -*-
# vi: set ft=ruby :

$install_ansible = <<-ANSIBLE
sudo apt install software-properties-common -y
sudo add-apt-repository --yes --update ppa:ansible/ansible -y
sudo apt install ansible -y
ANSIBLE

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

  if Vagrant.has_plugin? "vagrant-vbguest"
    config.vbguest.no_install = true
    config.vbguest.auto_update = false
    config.vbguest.no_remote = true
  end

  config.vm.define "manager" do |manager|
    manager.vm.box = "bento/ubuntu-20.04"
    manager.vm.hostname = "manager.local.io"
    manager.vm.network "private_network", ip: "192.168.56.2"
    manager.vm.provision "shell", inline: $install_ansible
    manager.vm.provision "ansible_local" do |ansible|
      ansible.playbook = "master-playbook.yml"
    end
  end
end
~
```

Puppet Provisioner

- En Vagrant, permite aprovisionar al guest usando Puppet, específicamente llamando a `puppet apply`, sin un Puppet Master
- Se necesita instalar puppet en la maquina guest
- Se debe especificar en Vagrant el uso de puppet como provisioner

```
Vagrant.configure("2") do |config|  
  config.vm.provision "puppet"  
end
```

Puppet – Estructura Mínima

- Por defecto, Vagrant configurará Puppet para buscar un manifests en la carpeta "manifests" relativa a la raíz del proyecto, y utilizará el manifest "default.pp" como punto de entrada

```
$ tree
.
|-- Vagrantfile
|-- manifests
|   |-- default.pp
```

Puppet – Módulos

- Vagrant también admite el aprovisionamiento con módulos Puppet. Esto se hace especificando una ruta a una carpeta de módulos. El archivo de manifest todavía se usa como punto de entrada.

```
Vagrant.configure("2") do |config|  
  config.vm.provision "puppet" do |puppet|  
    puppet.module_path = "modules"  
  end  
end
```


Personalizando Puppet

Basado en <https://github.com/patrickdlee/vagrant-examples>

```
# -*- mode: ruby -*-
# vi: set ft=ruby :

$install_puppet = <<-PUPPET
sudo apt-get install -y puppet
PUPPET

Vagrant.configure("2") do |config|
  config.vm.box = "bento/ubuntu-20.04"
  config.vm.hostname = "puppetServer"
  config.vm.network :private_network, ip: "192.168.90.3"
  config.vm.provision "shell", inline: $install_puppet
  config.vm.provision :puppet do |puppet|
    puppet.manifests_path = "puppet/manifests"
    puppet.manifest_file = "site.pp"
    puppet.module_path = "puppet/modules"
  end
end
```



Personalizando Puppet - Estructura

```
omondragon@Oscars-MacBook-Pro test_puppet % tree
```

```
.
├── Vagrantfile
└── puppet
    ├── manifests
    │   └── site.pp
    └── modules
        ├── baseconfig
        │   ├── files
        │   │   └── index.html
        │   ├── manifests
        │   └── init.pp
```

Personalizando Puppet – site.pp

```
include baseconfig
```

```
omondragon@Oscars-MacBook-Pro test_puppet % tree
.
├── Vagrantfile
└── puppet
    ├── manifests
    │   └── site.pp
    └── modules
        └── baseconfig
            ├── files
            │   ├── index.html
            │   └── manifests
            │       └── init.pp
```

Personalizando Puppet – index.html

```
<!DOCTYPE html>
<html>
<body>

<h1>Aprovisionando con Puppet</h1>

<p>Probando puppet</p>

</body>
</html>
```

```
omondragon@Oscars-MacBook-Pro test_puppet % tree
.
├── Vagrantfile
└── puppet
    ├── manifests
    │   └── site.pp
    └── modules
        ├── baseconfig
        │   ├── files
        │   │   ├── index.html
        │   └── manifests
        │       └── init.pp
```

Personalizando Puppet – init.pp

```
class baseconfig {  
  exec { 'apt-get update':  
    command => '/usr/bin/apt-get update';  
  }  
  
  package { ['apache2', 'tree']:  
    ensure => present;  
  }  
  
  file { '/var/www/html/index.html':  
    ensure => present,  
    owner  => 'root',  
    group  => 'root',  
    mode   => '0644',  
    source => 'puppet:///modules/baseconfig/index.html',  
    path   => '/var/www/html/index.html';  
  }  
  
  service { "apache2":  
    ensure => running,  
    enable => true,  
    require => Package['apache2'];  
  }  
}
```

```
}  
"modules/baseconfig/manifests/init.pp" 24L, 496C
```

```
omondragon@Oscars-MacBook-Pro test_puppet % tree
```

```
.  
├── Vagrantfile  
└── puppet  
    ├── manifests  
    │   └── site.pp  
    └── modules  
        └── baseconfig  
            ├── files  
            │   └── index.html  
            └── manifests  
                └── init.pp
```

Recursos Puppet

- Tipos de Recursos
 - File
 - Package
 - Service

Referencias

- Puppet. <https://puppet.com/>
- Vagrant Provisioning with Puppet. <https://medium.com/@Joachim8675309/vagrant-provisioning-with-puppet-553a59f0c48e>
- Automation testing with ansible, molecule and Vagrant. <https://www.trustedsec.com/blog/automation-testing-with-ansible-molecule-and-vagrant/>
- Correr Ansible en Windows. https://docs.ansible.com/ansible/2.5/user_guide/windows_faq.html
- Instalar WSL en Windows. <https://docs.microsoft.com/en-us/windows/wsl/install-win10>
- Aprovechamiento en Vagrant: <https://www.vagrantup.com/docs/provisioning>
- Vagrant Provisioning with Puppet. <https://medium.com/@Joachim8675309/vagrant-provisioning-with-puppet-553a59f0c48e>
- G.ST.02 Guía de Computación en la nube (2018): https://estrategia.gobiernoonlinea.gov.co/623/articles-75246_recurso_2.pdf