Установка и настройка vagrant

Введение

Время - деньги! © Уверен, что каждый слышал эту крылатую фразу знаменитого американского ученого и политического деятеля Бенджамена Франклина. В одной из предыдущих статей мы познакомились с десктопной системой виртуализации, которая облегчила нам жизнь, но на данный момент и этого уже не хватает. Так что, мы будем развивать нашу мысль дальше и познакомимся с прекрасным продуктом - vagrant, который позволит нам сэкономить время и облегчить разработку и тестирование продукта.

Не знаю почему, но во многих статьях в интернете указывается, что vagrant это надстройка поверх VirtualBox, на самом деле vagrant может использовать VirtualBox, VMware, Docker/lxc, Hyper-V в качестве т.н. провайдеров. Просто VirtualBox установлен провайдером по умолчанию, возможно так сложилось исторически.

Чтобы понять зачем же нам собственно Vagrant, давайте рассмотрим два варианта - AS IS и TO BE.

AS IS

Мы по старинке для каждого нового проекта создаем новую виртуальную машину в VirtualBox. Для этого нужно создать саму виртуальную машину, указать для неё необходимые параметры (размер и тип диска, количество оперативной памяти), после этого скачать и подключить установочный диск с операционной системой, запустить установку, дождаться её завершения, отключить установочный диск и перезагрузиться. А если виртуальная машина по каким-то причинам будет удалена, то всё это придётся делать заново. После успешной установки ОС необходимо установить и настроить среду разработки, будь то LAMP стек или что то другое. Даже у системного администратора это занимает много времени, что уже говорить о php/ruby/python разработчиках.

TO BE

Подразумевается, что мы используем vagrant. Потребуется всего несколько шагов

1. Клонируем код из системы контроля версий, будь то git/svn/mercurial. На самом деле сам Vagrantfile и все, что необходимо для настройки системы (provisioning) вы можете хранить где угодно, но если ваша команда разработчиков уже использует, например, git, то логично использовать именно ее. Но это уже как говорится на ваше усмотрение.

```
$ git clone git@bitbucket.org:projects/project1.git ~/projects/project1
```

2. Выполняем команду создания виртуальной машины

```
$ vagrant up
```

Идем пить кофе, в зависимости от скорости интернет каналов, а так же от производительности вашего компьютера операция может занять 10-20 минут и более. По сути это все! После успешного выполнения данной команды, у разработчика на компьютере будет

поднята и настроена готовая к использованию среда для разработки и тестирования.

Конечно, стоит уточнить, что vagrant не настроит магическим методом за вас виртуальную машину, он лишь позволит автоматизировать и унифицировать этот процесс, за счет использование промышленных систем управления и конфигурацией ПО, таких как chef, ansible, puppet, salt. Если вы не знакомы с этими системами, ничего страшного, можно использовать обычные shell скрипты.

Первое знакомство

И все было замечательно до того момента, когда ко мне обратились разработчики с проблемой не возможности запуска Vagrant на их машинах. Так как это front-end developers, то в качестве ОС они используют windows, так как 99% времени они проводят в Photoshop, Sublime и т.п. вещах, которых просто нет под Linux. Да, я знаю про MacOS, но, во-первых это дорого финансово, во-вторых переучивать людей и искать программы аналоги.

В итоге выяснилось, что заказчик в своем рабочем процессе (workflow) использует vagrant, а настройку машин (provisioning) производит с помощью ansible. И тут меня поджидала первая неприятность, так как на сайте ansible говорится

Control Machine Requirements

Currently Ansible can be run from any machine with Python 2.6 installed (**Windows isn’t supported for the control machine**).

Но так же в их блоге есть и такая информация

Initial windows support is now available on the development branch, and should be considered alpha-level but quite usable, and will evolve rapidly, with the expectation of being a beta release when Ansible 1.7 is released in a few months.

Ansible Windows modules are all written in PowerShell, and pushed out from a Linux control host using the excellent «winrm» library for Python.

Так что не все так плохо, как может показаться на первый взгляд.

Собственно, меня попросили изучить данный вопрос, а именно можно ли все таки запустить ansible на windows, так как мы не можем попросить заказчика изменить его workflow и перейти, например, с ansible на puppet. Но мы ведь так просто не сдаемся, так что приступаем к исследованию.

Подготовка окружения

В моем распоряжении был компьютер с ОС Windows 7 Pro SP1 x86, на который нам необходимо будет установить следующее ПО

1. Vagrant, на момент написания статьи это была следующая версия

```
C:\Users\borodach> vagrant --version
Vagrant 1.7.2
```

2. VirtualBox

```
C:\Users\borodach> VBoxManage.exe --version
4.3.22r98236
```

3. Cygwin

```
$ uname -a
CYGWIN_NT-6.1 PC1335 1.7.34(0.285/5/3) 2015-02-04 12:12 i686 Cygwin
```

При установке cygwin выбираем следующие пакеты

- python
- python-crypto
- python-paramiko
- python-setuptools
- wget
- openssh
- gcc-g++
- git
- mc

Установка ansible

Для установки ansible нам сначала необходимо установить pip - менеджер для управления модулями python, некий аналог CPAN/yum.

Запускаем cygwin консоль, и производим установку pip

```
$ /usr/bin/easy_install pip
$ pip install ansible
```

После успешной установки всех зависимостей и самого ansible мы должны получить примерно такой список модулей

```
$ pip list
ansible (1.8.3)
distribute (0.6.34)
ecdsa (0.13)
Jinja2 (2.7.3)
MarkupSafe (0.23)
paramiko (1.15.2)
pip (6.0.8)
pycrypto (2.6.1)
```

```
PyYAML (3.11)
setuptools (0.6rc11)
```

Тестирование

Как я уже писал в начале статьи, Vagrantfile, а так же все скрипты/yaml файлы и playbooks удобно хранить в системе контроля версий, что наш заказчик и делает. Поэтому клонируем уже готовый репозитарий

```
$ mkdir -p ~/clientID/projects/project1
$ cd ~/clientID/projects/project1
$ git clone --recursive git@bitbucket.org:clientID/ansible-roles.git ./
```

В результате в папке project1 у нас располагаются следующие файлы

```
$ ls
data  playbook.yml  README.md  roles  Vagrantfile  vars
```

Теперь все готово к запуску виртуальной машины. Пробуем.

```
$ vagrant up
Bringing machine 'default' up with 'virtualbox' provider...
==> default: Importing base box 'ubuntu/trusty64'...
==> default: Matching MAC address for NAT networking...
==> default: Checking if box 'ubuntu/trusty64' is up to date...
==> default: Setting the name of the VM:
project1_default_1424444952969_22189
==> default: Clearing any previously set forwarded ports...
==> default: Fixed port collision for 22 => 2222. Now on port 2201.
==> default: Clearing any previously set network interfaces...
==> default: Preparing network interfaces based on configuration...
    default: Adapter 1: nat
    default: Adapter 2: hostonly
==> default: Forwarding ports...
    default: 22 => 2201 (adapter 1)
==> default: Running 'pre-boot' VM customizations...
==> default: Booting VM...
==> default: Waiting for machine to boot. This may take a few minutes...
    default: SSH address: 127.0.0.1:2201
    default: SSH username: vagrant
    default: SSH auth method: private key
    default: Warning: Connection timeout. Retrying...
    default: Warning: Connection timeout. Retrying...
    default: Warning: Remote connection disconnect. Retrying...
    default: Warning: Remote connection disconnect. Retrying...
    default:
    default: Vagrant insecure key detected. Vagrant will automatically
```

```
replace
    default: this with a newly generated keypair for better security.
    default:
    default: Inserting generated public key within guest...
    default: Removing insecure key from the guest if its present...
    default: Key inserted! Disconnecting and reconnecting using new SSH
key...
==> default: Machine booted and ready!
==> default: Checking for guest additions in VM...
==> default: Configuring and enabling network interfaces...
==> default: Mounting shared folders...
    default: /var/www =>
C:/cygwin/home/borodach/clientID/projects/project1/www
    default: /vagrant => C:/cygwin/home/borodach/clientID/projects/project1
==> default: Running provisioner: ansible...
PYTHONUNBUFFERED=1 ANSIBLE_FORCE_COLOR=true ANSIBLE_HOST_KEY_CHECKING=false
ANSIBLE_SSH_ARGS='-o UserKnownHostsFile=/dev/null -o ForwardAgent=yes -o
ControlMaster=auto -o ControlPersist=60s' ansible-playbook --private-
key=C:/cygwin/home/borodach/clientID/projects/project1/.vagrant/machines/def
ault/virtualbox/private_key --user=vagrant --connection=ssh --
limit='default' --inventory-
file=C:/cygwin/home/borodach/clientID/projects/project1/.vagrant/provisioner
s/ansible/inventory --sudo playbook.yml
The executable 'ansible-playbook' Vagrant is trying to run was not
found in the %PATH% variable. This is an error. Please verify
this software is installed and on the path.
```

И здесь нас ждет первая неудача, vagrant не видит ansible/ansible-playbook, хотя они и находятся в переменных окружения

```
$ echo $PATH
/usr/local/bin:/usr/bin:/cygdrive/c/Windows/system32:/cygdrive/c/Windows:/cy
gdrive/c/Windows/System32/Wbem:/cygdrive/c/Windows/System32/WindowsPowerShel
l/v1.0:/cygdrive/c/HashiCorp/Vagrant/bin

$ whereis ansible
ansible: /usr/bin/ansible

$ whereis ansible-playbook
ansible-playbook: /usr/bin/ansible-playbook
```

Решение нашел на этом сайте. Т.е. нам необходимо создать bat файл и положить его в любой путь, который прописан в PATH на windows, я для этого выбрал C:\HashiCorp\Vagrant\bin. Создаем файл ansible-playbook.bat с таким содержимым

```
@echo off

set CYGWIN=C:\cygwin
set SH=%CYGWIN%\bin\bash.exe
"%SH%" -c "/bin/ansible-playbook %*"
```

Так как сама виртуальная машина уже поднята, в чем можно убедиться

```
$ vagrant status
Current machine states:

default                   running (virtualbox)

The VM is running. To stop this VM, you can run `vagrant halt` to
shut it down forcefully, or you can run `vagrant suspend` to simply
suspend the virtual machine. In either case, to restart it again,
simply run `vagrant up`.
```

У нас не отработала лишь стадия provisioning, то можно запускать просто следующую команду - vagrant provision. И снова нас ждет неудача

```
$ vagrant provision
==> default: Running provisioner: ansible...
PYTHONUNBUFFERED=1 ANSIBLE_FORCE_COLOR=true ANSIBLE_HOST_KEY_CHECKING=false
ANSIBLE_SSH_ARGS='-o UserKnownHostsFile=/dev/null -o ForwardAgent=yes -o
ControlMaster=auto -o ControlPersist=60s' ansible-playbook --private-
key=C:/cygwin/home/borodach/clientID/projects/project1/.vagrant/machines/def
ault/virtualbox/private_key --user=vagrant --connection=ssh --
limit='default' --inventory-
file=C:/cygwin/home/borodach/clientID/projects/project1/.vagrant/provisioner
s/ansible/inventory --sudo playbook.yml

PLAY [all]
*********************************************************************

GATHERING FACTS
*********************************************************************
fatal: [default] => private_key_file
(C:/cygwin/home/borodach/clientID/projects/project1/.vagrant/machines/defaul
t/virtualbox/private_key) is group-readable or world-readable and thus
insecure - you will probably get an SSH failure

TASK: [Ensure apt-get is uptodate]
****************************************
FATAL: no hosts matched or all hosts have already failed -- aborting


PLAY RECAP
********************************************************************
           to retry, use: --limit @/home/borodach/playbook.retry

default                    : ok=0    changed=0    unreachable=1    failed=0

Ansible failed to complete successfully. Any error output should be
visible above. Please fix these errors and try again.
```

Опять идем в google и находим вот этот открытый баг. Всякие пляски с chmod, ntfs правами на private key мне не помогли, похоже это ограничение самой среды cygwin. Так что для обхода этого неудобства пришлось изменить файл C:\cygwin\lib\python2.7\site-packages\ansible\runner\connection.py. Я просто удалил проверку прав на private key

connection.py

Wrap text | Diff comments | View file @9f31e03

```
...  ...  @@ -22,20+22,20 @@
22   22   import stat
23   23   import errno
24   24
25   25   from ansible import utils
26   26   from ansible.errors import AnsibleError
27   27
28   28   class Connector(object):
29   29       ''' Handles abstract connections to remote hosts '''
30   30
31   31       def __init__(self, runner):
32   32           self.runner = runner
33   33
34   34       def connect(self, host, port, user, password, transport, private_key_file):
35   35           conn = utils.plugins.connection_loader.get(transport, self.runner, host, port, user=user, password=password, private_key_file=private_key_file)
36   36           if conn is None:
37   37               raise AnsibleError("unsupported connection type: %s" % transport)
38   38           if private_key_file:
39   39               # If private key is readable by user other than owner, flag an error
40   40               st = None
41   41               try:
42   42                   st = os.stat(private_key_file)
43   43               except (IOError, OSError), e:
44   44                   if e.errno != errno.ENOENT: # file is missing, might be agent
45   45                       raise(e)
46        -
47        -              if st is not None and st.st_mode & (stat.S_IRGRP | stat.S_IROTH):
48        -                  raise AnsibleError("private_key_file (%s) is group-readable or world-readable and thus insecure - "
49        -                                     "you will probably get an SSH failure"
50        -                                     % (private_key_file,))
51   46          self.active = conn.connect()
52   47          return self.active
...  ...
```

Снова запускаем vagrant provision. И на этот раз успех, ansible отрабатывает успешно.

```
$ vagrant provision
==> default: Running provisioner: ansible...
PYTHONUNBUFFERED=1 ANSIBLE_FORCE_COLOR=true ANSIBLE_HOST_KEY_CHECKING=false
ANSIBLE_SSH_ARGS='-o UserKnownHostsFile=/dev/null -o ForwardAgent=yes -o
ControlMaster=auto -o ControlPersist=60s' ansible-playbook --private-
key=C:/cygwin/home/borodach/clientID/projects/project1/.vagrant/machines/def
ault/virtualbox/private_key --user=vagrant --connection=ssh --
limit='default' --inventory-
file=C:/cygwin/home/borodach/clientID/projects/project1/.vagrant/provisioner
s/ansible/inventory --sudo playbook.yml

PLAY [all]
*********************************************************************

GATHERING FACTS
*********************************************************************
ok: [default]

TASK: [Ensure apt-get is uptodate]
*****************************************
ok: [default]
```

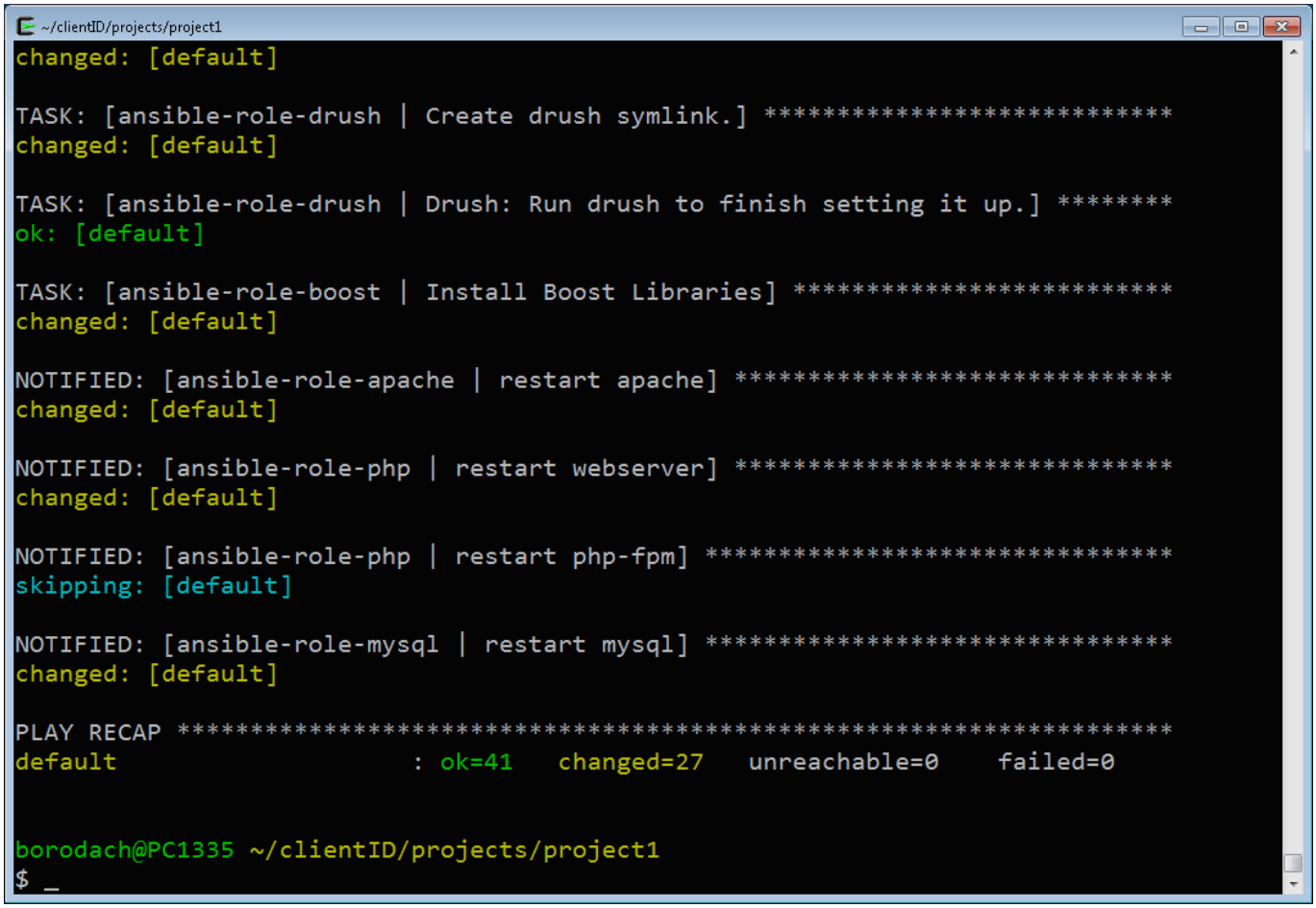
```
TASK: [ansible-role-apache | Ensure Apache is installed.]
*********************
changed: [default] => (item=apache2)

TASK: [ansible-role-apache | Configure Apache.]
*****************************
ok: [default] => (item={'regexp': '^Listen ', 'line': u'Listen 80'})
...
...
...
NOTIFIED: [ansible-role-php | restart webserver]
*****************************
changed: [default]

NOTIFIED: [ansible-role-php | restart php-fpm]
*******************************
skipping: [default]

NOTIFIED: [ansible-role-mysql | restart mysql]
*******************************
changed: [default]

PLAY RECAP
********************************************************************
default                    : ok=41   changed=27   unreachable=0    failed=0
```



Уже появился соответствующий pull request для исправления данной проблемы на cygwin.

Для общей информации привожу содержимое Vagrantfile

```
$ cat Vagrantfile
VAGRANTFILE_API_VERSION = "2"
Vagrant.configure(VAGRANTFILE_API_VERSION) do |config|
  config.vm.box = "ubuntu/trusty64"
  config.vm.network "private_network", ip: "192.168.33.99"
  config.ssh.forward_agent = true
  config.vm.synced_folder "data", "/vagrant/data", id: "vagrant-root",
    mount_options: ["dmode=775,fmode=664"],
    create: true

  config.vm.synced_folder "www", "/var/www",
    id: "vagrant-root",
    create: true,
    owner: 'vagrant',
    group: 'www-data',
    mount_options: ["dmode=775,fmode=775"]

  config.vm.provider "virtualbox" do |vb|
    vb.customize ["modifyvm", :id, "--memory", "512"]
  end

  # Provisioning configuration for Ansible.
```

```
  config.vm.provision "ansible" do |ansible|
    ansible.playbook = "playbook.yml"
    ansible.sudo = true
  end
end
```

Подводные камни

В процессе настройки данной связки столкнулся с двумя проблемами. Во время выполнения команд vagrant в произвольном порядке выпадала ошибка

```
GATHERING FACTS
*****************************************************************
ok: [default]

TASK: [Ensure apt-get is uptodate]
*******************************************
ok: [default]

TASK: [ansible-role-apache | Ensure Apache is installed.]
*********************
      0 [main] python2.7 2820 child_info_fork::abort: address space needed
by 'cygcrypto-1.0.0.dll' (0x8D0000) is already occupied
```

Причем каждый раз библиотека могла быть разной. Связанно с тем, что windows не умеет делать fork/exec как это принято во всех Unix системах.

Unfortunately, Windows does not use the fork/exec model of process creation found in UNIX-like OSes, so it is difficult for Cygwin to implement a reliable and correct fork()

Мне помогло выполнение rebase

Вторая проблема, при попытке выполнить provisioning получал ошибку

```
GATHERING FACTS
*****************************************************************
fatal: [app] => SSH encountered an unknown error during the connection. We
recommend you re-run the command using -vvvv, which will enable SSH
debugging output to help diagnose the issue
```

Как и советуют выше, передаем параметр -vvvv в ansible, для этого достачно добавить ansible.verbose в описание нашего provision. В результате должно получиться следующее

```
config.vm.provision "ansible" do |ansible|
   ansible.playbook = "playbook.yml"
   ansible.sudo = true
   ansible.verbose = "vvvv"
end
```

Снова запускаем vagrant provision, но на этот раз мы получим очень много информации, так как ssh соеденниние будет запущенно в режиме отладки. Обращаем внимение на следующие строки

```
debug2: process_mux_new_session: channel 1: request tty 1, X 0, agent 1,
subsys 0, term "xterm", cmd "/bin/sh -c 'mkdir -p
$HOME/.ansible/tmp/ansible-tmp-1424598908.92-233069445915350 && chmod a+rx
$HOME/.ansible/tmp/ansible-tmp-1424598908.92-233069445915350 && echo
$HOME/.ansible/tmp/ansible-tmp-1424598908.92-233069445915350'", env 0
debug3: mm_receive_fd: recvmsg: Resource temporarily unavailable
mm_receive_fd: no message header
process_mux_new_session: failed to receive fd 0 from slave
debug1: channel 1: mux_rcb failed
debug2: channel 1: zombie
debug2: channel 1: gc: notify user
```

Гугл нам подсказывает, что проблема в ControlMaster, его необходимо выключить. Для этого добавляем параметр ansible.raw_ssh_args в наш Vagrantfile

```
config.vm.provision "ansible" do |ansible|
   ansible.playbook = "playbook.yml"
   ansible.sudo = true
   ansible.verbose = "vvvv"
   ansible.raw_ssh_args = "-o ControlMaster=no"
end
```

И пробуем снова, на этот раз все отрабатывает без ошибок. После этого можно убрать параметр ansible.verbose, чтобы не засорять вывод лишней информацией.

Так же, я нашел еще один способ решения проблемы - для этого необходимо создать файл .ansible.cfg в домашней папке пользователя с таким содержимым

```
$ cat ~/.ansible.cfg
[ssh_connection]
control_path = /tmp
```