Create a Simple Ansible Test Environment

Ansible runs on Linux or Mac OSX and requires several third-party libraries, making it impossible to test Ansible on Windows workstations.

This document describes a simple process you can use to install Ansible in an Ubuntu virtual machine created with Vagrant and running within VirtualBox on your workstation/laptop.

You can create the test environment in numerous other ways, including using Python virtual environment in OSX, Docker containers, other Linux distributions, other hypervisor products, or other VM provisioning system. These alternatives are beyond the scope of this document.

Release Notes

June 20th 2017:

- VMware Fusion/Workstation required if you want to use VIRL
- Added information on Vagrant providers

August 10th 2017:

- Added **apt-get update** to refresh APT package lists.

October 26th 2018:

- Migrated to Ubuntu 18.04

Download and Install Prerequisite Software

Download and install:

- VirtualBox or VMware Workstation/Fusion
- Vagrant

If you want to use your Ansible VM with Cisco's VIRL use VMware Workstation or Fusion instead of VirtualBox. VIRL does not work within VirtualBox and it's impossible to get VirtualBox VM communicating with VMware virtual networks used by VIRL.

Create an Ubuntu Virtual Machine

After installing VirtualBox and Vagrant you'll create an Ubuntu virtual machine on which you'll later install Python and Ansible:

- Create an empty directory
- Open a terminal window and navigate to the directory you just created
- Create simple Vagrant environment using **vagrant init** command.

```
$ vagrant init bento/ubuntu-18.04
A `Vagrantfile` has been placed in this directory. You are now
ready to `vagrant up` your first virtual environment! Please read
the comments in the Vagrantfile as well as documentation on
`vagrantup.com` for more information on using Vagrant.
```

I'm using **bento** boxes as they support all four major Vagrant providers (VirtualBox, VMware Workstation and Fusion, Hyper-V). You can use **ubuntu** boxes, but they won't run on anything but Virtualbox.

- Start the Ubuntu virtual machine using VirtualBox virtualization with **vagrant up --provider virtualbox** command.

Start the VM with **vagrant up –provider vmware_fusion** (or **vmware_workstation**) if you use VMware Fusion/Workstation

```
$ vagrant up --provider virtualbox
Bringing machine 'default' up with 'virtualbox' provider...
==> default: Box 'bento/ubuntu-18.04' could not be found. Attempting to
find and install...
```

```
    default: Box Provider: virtualbox
    default: Box Version: >= 0
==> default: Loading metadata for box 'bento/ubuntu-18.04'
…
==> default: Successfully added box 'bento/ubuntu-18.04' (v201808.24.0)
for 'virtualbox'!
==> default: Importing base box 'bento/ubuntu-18.04'...
==> default: Matching MAC address for NAT networking...
==> default: Checking if box 'bento/ubuntu-18.04' is up to date...
==> default: Setting the name of the VM: test_default_1540556166911_50860
==> default: Clearing any previously set network interfaces...
==> default: Preparing network interfaces based on configuration...
    default: Adapter 1: nat
==> default: Forwarding ports...
    default: 22 (guest) => 2222 (host) (adapter 1)
==> default: Booting VM...
==> default: Waiting for machine to boot. This may take a few minutes...
    default: SSH address: 127.0.0.1:2222
    default: SSH username: vagrant
    default: SSH auth method: private key
    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 it's present...
    default: Key inserted! Disconnecting and reconnecting using new SSH
key...
==> default: Machine booted and ready!
==> default: Mounting shared folders...
    default: /vagrant => /Users/ip/test
```

You might get tons of additional printouts if Vagrant decides it's time to install a new version of VirtualBox Guest Additions.

- Log into the virtual machine using **vagrant ssh** command:

```
$ vagrant ssh
Welcome to Ubuntu 18.04.1 LTS (GNU/Linux 4.15.0-29-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage
[…]
vagrant@vagrant:~$
```

Notes:

- You can shut down the virtual machine with **vagrant halt** command
- The next time you want to bring up the virtual machine use **vagrant up** *in the same directory*.

INSTALL GIT ON UBUNTU VM

In this step you'll install **git** client on the Ubuntu VM you just created. Git is usually bundled with the Vagrant box you're using, so might be able to skip this step.

- Log into the Ubuntu VM with **vagrant ssh**
- You might want to change the default *bash* prompt:

```
vagrant@vagrant:~$ export PS1="\W$ "
```

- Refresh software package index files with **sudo apt-get update** command
- Make sure *git* is installed with **sudo apt-get -qq install git**.

Notes:

- **sudo** command temporarily makes you the **root** user (system administrator). In virtual machines set up by Vagrant the **vagrant** user can always execute the **sudo** command.
- **apt-get** command installs new software packages.
- **-qq** option of **apt-get** command reduces the clutter printed by **apt-get** to the bare minimum.

Clone the Ansible Examples

To clone the Ansible/Python/Jinja2 examples from the ipSpace.net GitHub repository use these steps:

- Log into the Ubuntu VM with **vagrant ssh**
- Change working directory to /vagrant

```
~$ cd /vagrant
```

- Clone the git repository with **git clone** command

```
vagrant$ git clone https://github.com/ipspace/NetOpsWorkshop.git
Cloning into 'NetOpsWorkshop'...
remote: Counting objects: 382, done.
remote: Total 382 (delta 0), reused 0 (delta 0), pack-reused 382
Receiving objects: 100% (382/382), 70.60 KiB | 0 bytes/s, done.
Resolving deltas: 100% (129/129), done.
Checking connectivity... done.
```

Notes:

- The **/vagrant** directory on Ubuntu virtual machine is mapped to the host directory in which you created the Vagrant environment with **vagrant init**. Any changes you make in that directory or its subdirectories in Ubuntu VM automatically appear in your host file system (and vice versa).
- Host file system and Ubuntu VM use file locking to prevent unwanted changes to the file system. For example, if you change into a subdirectory of your Vagrant directory in a terminal window on your host system, or if you a folder window opened in a subdirectory of your Vagrant directory, you cannot remove that subdirectory with **rmdir** command in the Ubuntu VM (and vice versa).
- The **git clone** command copies the contents of GitHub repository into the **NetOpsWorkshop** subdirectory within your Vagrant directory. You can change the subdirectory name by specifying a name after the URL in **git clone** command or by using **mv** command.

Install Additional Software on Your Ubuntu VM

In the last step you'll install additional Ubuntu software packages, Python libraries, and Ansible on your Ubuntu VM:

- Log into your Ubuntu VM with **vagrant ssh**
- Navigate into the **install** subdirectory of the cloned GitHub repository:

```
~$ cd /vagrant/NetOpsWorkshop/install/
```

- Run the **install** *bash* script:

```
install$ ./install.sh
Update installed software to latest release
Install missing packages
…
```

Explore

You'll find examples from the *Ansible for Networking Engineers* webinar in these directories on your Ubuntu VM:

Examples	Directory
YAML	/vagrant/NetOpsWorkshop/YAML
Jinja2	/vagrant/NetOpsWorkshop/Jinja2
Jinja2 ipaddr filters	/vagrant/NetOpsWorkshop/Jinja2/ipaddr
Ansible (basics)	/vagrant/NetOpsWorkshop/Ansible
Ansible networking modules	/vagrant/NetOpsWorkshop/Ansible/Networking

Connect to Network Devices

This document does not cover more extensive virtual lab setups, but I'm positive you'll find plenty of useful examples on the 'net (we also cover them in the Building Network Automation Solutions online course).

If you're not familiar with virtual lab setups or tools used in this document, it might be easiest to connect to a physical device – as long as the device is reachable from your workstation, it's also reachable from the Ubuntu VM due to the automatic outbound NAT set up by Vagrant.