Appendix A - Using Ansible on Windows workstations

Ansible works primarily over the SSH protocol, which is supported natively by most every server, workstation, and operating system on the planet, with one exception—Microsoft's venerable Windows OS (though this may change in the coming years).

To use SSH on Windows, you need additional software. But Ansible also requires other utilities and subsystems only present on Linux or other UNIX-like operating systems. This poses a problem for many system administrators who are either forced to use or have chosen to use Windows as their primary OS.

This appendix will guide Windows users through the author's preferred method of using Ansible on a Windows workstation.

Ansible 1.7 and later can manage Windows hosts (see Ansible's Windows Support[198] documentation), but doesn't run within Windows natively. You still need to follow the instructions here to run the Ansible client on a Windows host.

Method 1 - Use the Windows Subsystem for Linux / Bash on Ubuntu

If you are running Windows 10, and have installed either the Anniversary Update or any later version, you can install the Windows Subsystem for Linux (WSL), which is the most seamless Bash integration you can currently get for Windows.

The WSL downloads Ubuntu and places it in a special privileged VM layer that's as transparent as it can be while still existing sandboxed from the general Windows

[198] http://docs.ansible.com/intro_windows.html

environment. Using WSL, you can open up an Ubuntu command prompt and have access to almost all the same software and functionality you would have if you were running Ubuntu natively!

Microsoft has the most up-to-date installation guide[199] on their Developer Network site, but the installation process is straightforward:

1. Turn on Developer mode (inside Settings > Update and Security > For developers).
2. Open a PowerShell prompt as an administrator and run the command:

```
Enable-WindowsOptionalFeature -Online -FeatureName \
Microsoft-Windows-Subsystem-Linux
```

3. Restart your computer when prompted.

At this point, the WSL is installed, but Ubuntu has not yet been installed. To do that:

1. Open a Command prompt (`cmd`), and run the command `bash`.
2. Accept the license by typing `y` when prompted.
3. The first time Ubuntu is installed, you'll also be asked for a username and password to use in the bash environment.

Once installation completes, there will be a shortcut either on your Desktop or in the Start menu, and you can either use this shortcut to launch a `bash` session, or type `bash` in a Command prompt.

Now that you have Bash on Ubuntu running inside Windows, you can install Ansible inside the WSL environment just like you would if you were running Ubuntu natively!

Installing Ansible inside Bash on Ubuntu

Before installing Ansible, make sure your package list is up to date by updating apt-get:

[199] https://msdn.microsoft.com/en-us/commandline/wsl/install_guide

```
$ sudo apt-get update
```

The easiest way to install Ansible is to use `pip`, a package manager for Python. Python should already be installed on the system, but `pip` may not be, so let's install it, along with Python's development header files (which are in the `python-dev` package).

```
$ sudo apt-get install -y python-pip python-dev
```

After the installation is complete, install Ansible:

```
$ sudo pip install ansible
```

After Ansible and all its dependencies are downloaded and installed, make sure Ansible is running and working:

```
$ ansible --version
ansible 2.9.5
```

Upgrading Ansible is also easy with pip: Run `sudo pip install --upgrade ansible` to get the latest version.

You can now use Ansible within the Ubuntu Bash environment. You can access files on the Windows filesystem inside the `/mnt` folder (`/mnt/c` corresponds to `C:\`), but be careful when moving things between Windows and the WSL, as strange things can happen because of line ending, permissions, and filesystem differences!

Method 2 - When WSL is not an option

If you're running Windows 7 or 8, or for some reason can't install or use the Windows Subsystem for Linux in Windows 10 or later, then the best alternative is to build a local Virtual Machine (VM) and install and use Ansible inside.

Prerequisites

The easiest way to build a VM is to download and install Vagrant and VirtualBox (both 100% free!), and then use Vagrant to install Linux, and PuTTY to connect and use Ansible. Here are the links to download these applications:

1. Vagrant[200]
2. VirtualBox[201]
3. PuTTY[202]

Once you've installed all three applications, you can use either the command prompt (`cmd`), Windows PowerShell, or a Linux terminal emulator like Cygwin to boot up a basic Linux VM with Vagrant (if you use Cygwin, which is not covered here, you could install its SSH component and use it for SSH, and avoid using PuTTY).

Set up an Ubuntu Linux Virtual Machine

Open PowerShell (open the Start Menu or go to the Windows home and type in 'PowerShell'), and change directory to a place where you will store some metadata about the virtual machine you're about to boot. I like having a 'VMs' folder in my home directory to contain all my virtual machines:

```
# Change directory to your user directory.
PS > cd C:/Users/[username]
# Make a 'VMs' directory and cd to it.
PS > md -Name VMs
PS > cd VMs
# Make a 'Ubuntu64' directory and cd to it.
PS > md -Name ubuntu-bionic-64
PS > cd ubuntu-bionic-64
```

Now, use `vagrant` to create the scaffolding for our new virtual machine:

[200] http://www.vagrantup.com/downloads.html
[201] https://www.virtualbox.org/
[202] http://www.chiark.greenend.org.uk/~sgtatham/putty/download.html

```
PS > vagrant init ubuntu/bionic64
```

Vagrant creates a 'Vagrantfile' describing a basic Ubuntu 64-bit virtual machine in the current directory, and is now ready for you to run `vagrant up` to download and build the machine. Run `vagrant up`, and wait for the box to be downloaded and installed:

```
PS > vagrant up
```

After a few minutes, the box will be downloaded and a new virtual machine set up inside VirtualBox. Vagrant will boot and configure the machine according to the defaults defined in the Vagrantfile. Once the VM is booted and you're back at the command prompt, it's time to log into the VM.

Log into the Virtual Machine

Use `vagrant ssh-config` to grab the SSH connection details, which you will then enter into PuTTY to connect to the VM.

```
PS > vagrant ssh-config
```

It should show something like:

```
Host default
  Hostname 127.0.0.1
  User vagrant
  Port 2222
  UserKnownHostsFile /dev/null
  StrictHostKeyChecking no
  PasswordAuthentication no
  IdentityFile C:/Users/[username]/.vagrant.d/insecure_private_key
  IdentitiesOnly yes
  LogLevel FATAL
```

The lines we're interested in are the Hostname, User, Port, and IdentityFile.

Launch PuTTY, and enter the connection details:

- **Host Name (or IP address)**: 127.0.0.1
- **Port**: 2222

Click Open to connect, and if you receive a Security Alert concerning the server's host key, click 'Yes' to tell PuTTY to trust the host. You can save the connection details by entering a name in the 'Saved Sessions' field and clicking 'Save' to save the details.

PuTTY will ask for login credentials; we'll use the default login for a Vagrant box (`vagrant` for both the username and password):

```
login as: vagrant
vagrant@127.0.0.1's password: vagrant
```

You should now be connected to the virtual machine, and see the message of the day:

```
Welcome to Ubuntu 18.04.1 LTS (GNU/Linux 4.15.0-43-generic x86_64)
...
vagrant@ubuntu-bionic:~$
```

If you see this prompt, you're logged in, and you can start administering the VM. The next (and final) step is to install Ansible.

This example uses PuTTY to log into the VM, but other applications like Cygwin[203] or Git for Windows[204] work just as well, and may be easier to use. Since these alternatives have built-in SSH support, you don't need to do any extra connection configuration, or even launch the apps manually; just `cd` to the same location as the Vagrantfile, and enter `vagrant ssh`!

[203] http://cygwin.com/install.html
[204] http://git-scm.com/download/win

Install Ansible

Before installing Ansible, make sure your package list is up to date by updating apt-get:

```
$ sudo apt-get update
```

The easiest way to install Ansible is to use `pip`, a package manager for Python. Python should already be installed on the system, but `pip` may not be, so let's install it, along with Python's development header files (which are in the `python-dev` package).

```
$ sudo apt-get install -y python-pip python-dev
```

After the installation is complete, install Ansible:

```
$ sudo pip install ansible
```

After Ansible and all its dependencies are downloaded and installed, make sure Ansible is running and working:

```
$ ansible --version
ansible 2.9.5
```

Upgrading Ansible is also easy with pip: Run `sudo pip install --upgrade ansible` to get the latest version.

You should now have Ansible installed within a virtual machine running on your Windows workstation. You can control the virtual machine with Vagrant (`cd` to the location of the Vagrantfile), using `up` to boot or wake the VM, `halt` to shut down the VM, or `suspend` to sleep the VM. Log into the VM manually using PuTTY or via `vagrant ssh` with Cygwin or Git's Windows shell.

Use Ansible from within the virtual machine just as you would on a Linux or Mac workstation directly. If you need to share files between your Windows environment and the VM, Vagrant conveniently maps `/vagrant` on the VM to the same folder where your Vagrantfile is located. You can also connect between the two via other methods (SSH, SMB, SFTP etc.) if you so desire.

Summary

There are other ways to 'hack' Ansible into running natively within Windows (without a Linux VM), such as the ansible-babun-bootstrap[205], but I recommend either using the WSL or running everything within a Linux VM as performance will be optimal and the number of environment-related problems you encounter will be greatly reduced!

[205] https://github.com/jonathanhle/ansible-babun-bootstrap