Ansible & Vagrant

Objectives

- Build familiarity with Ansible and Vagrant
- Create a development environment
- Develop a simple role to configure a new environment
- Discuss how to easily use the same role for development and production

Who am I?

- Director of Infrastructure at Draconyx, LLC.
- Free and Open Source Software enthusiast
- Co-Founder of the Evansville Linux User Group
- Member of OpenNSM
- Contributor of sickbits.net

What is Ansible?

- Open-source configuration management and IT automation platform
- Comparable to Puppet, Chef, Salt, etc.
- Written in Python

Why Ansible?

- Agentless
- Minimal changes to infrastructure
- Easy to learn
- Easy to read (YAML syntax)
- Ability to write custom modules in any language
- Inventory of the tasks performed to set up machines

Ansible Terms

- Playbook – List of plays used for configuration
- Play – Matches hosts to roles
- Task – Call to a module to perform an action
- Module – Scripts copied to and ran on host
- Role – A specific grouping structure to allow for automatic inclusion of variables, tasks, and handlers

What is Vagrant?

- Open-Source tool used to provision virtual development environments
- Uses Ruby syntax

Why use Vagrant with Ansible?

- Quick set up and tear down of development environments
- Easy to rebuild if mistakes are made
- Test new configurations and changes
- Makes sharing ideas and systems easier
- SAVES YOUR TIME!

Setting up a Vagrant box for Development

- vagrant init
- Edit the Vagrantfile
- vagrant up
- vagrant ssh

Vagrant init

- Vagrant init *[boxname] [box_url]*

Vagrant up!

- New VM in moments
- Output will display SSH user, IP, and port.
- Using 'vagrant ssh' will connect the user to the Virtual Machine
  - Private key is located at ~/.vagrant.d/insecure_private_key

Other Vagrant commands

- Vagrant provision – run provisioner on the VM
- Vagrant status – View status of VMs
- Vagrant halt – Shutdown VM
- Vagrant reload – Reboot the VM
- Vagrant destroy – Remove the VM

The Ansible file structure

- Playbook.yml
- Roles/
  - Website/
    - Files/
    - Templates/

Ansible roles

- Makes sharing configurations easier
- Creates a structure that's easy to follow
- Allows for automatic inclusion of vars, tasks and handlers
- Separates complex playbooks into smaller files

Prepare your automation

- Create the inventory file
- Create the host_vars file
- Create a task list for the role
- Create a handlers file
- Include files to be copied
- Create playbook linking host to the role

Edit the Vagrantfile

- Set Ansible as the provisioner
- Specify the playbook and inventory
- Tell Ansible not to set the --limit flag

Vagrant Provision

- Provision the machine without rebooting
- Run the configuration against the environment
- Can be ran multiple times to test changes

Ready for production?

- Using roles, the configuration can easily ran against another host with the following steps:
  - Add the production machine to inventory
  - Create a new variable file in host_vars/
  - Create a new playbook linking the previous role to the production host

Questions?