Introduction to Ansible

A very brief introduction

Ansible

What is Ansible?

- According to the official website:
  - Ansible is a radically simple IT automation engine that automates cloud provisioning, configuration management, application deployment, intra-service orchestration, and many other IT needs.

What do I know about Ansible?

- Not too much – yet...
  - Easy to get started with
  - Requires python 2.6 or better
  - Only uses ssh for communications
  - You can run arbitrary commands across systems
  - You can create command sets/states for your servers called playbooks

How it works

- Ansible simply uses SSH to push changes from wherever it runs (a server or your own laptop)
- Conceptually, it's as if instead of connecting to your machines with SSH and running commands manually, you could script the whole thing and run it automatically
  - Not that I've ever done such a thing
    - (and admitted to it)

How it works (contd)

- It comes with better tools to script these connections, mostly modules that make it more convenient to do the most common operations and ensure idempotence[1]

[1] Ensuring that even if you run the same thing twice you'll end up in the same final state

Why would you want to use it?

- You can deploy master files to systems easily
- You can guarantee what version of software is running
- You can manage services and packages with simple commands
- You can make changes to ini files with some degree of confidence
- Etc.

Getting started…

- You need at least one machine to manage
  - It can be the machine ansible is running on
- You need python
- You need to install ansible on your management machine
- You need to be able to ssh into the target machine(s)

Ansible Installation

- Simple
  - Assuming a unix-ish base for this
  - Assuming you have a recent python installed

```
$ sudo easy_install pip
$ sudo pip install ansible
```

  - Test to make sure it is working

```
$ ansible –version
```

- You can use package managers as well
  - Yum, apt-get, etc.

Inventory

- Ansible wants an inventory to work from
  - This can just be a plain list of hosts
    - It can be divided into groups
    - It can have comments and other information
  - It can be manually created or dynamically created
    - Manually with your hosts file, a dns dump, etc.
    - Dynamically with components such as cobbler, openstack inventory, EC2 external inventory, etc.

Example Inventory

```
#
# This is the ansible 'hosts' file, usually found in
# /etc/ansible/hosts
#
[webservers]
node1
node2
node3
node4
[balancer]
node5
[db]
node6
```

First command: ping

- You should ensure you have the ability to reach your inventory

```
ansible all -m ping
```

Output

```
network@ansible:~$ ansible all -m
ping
node2 | SUCCESS => {
    "changed": false,
    "ping": "pong"
}
node6 | SUCCESS => {
    "changed": false,
    "ping": "pong"
}
node1 | SUCCESS => {
    "changed": false,
    "ping": "pong"
}
```

```
node3 | SUCCESS => {
    "changed": false,
    "ping": "pong"
}
node5 | SUCCESS => {
    "changed": false,
    "ping": "pong"
}
node4 | SUCCESS => {
    "changed": false,
    "ping": "pong"
}
```

How about installing software?

- Install nginx on all of the new web servers
  - How would we do it?

```
ansible webservers -s -m shell -a 'apt-get install
nginx'
```

Not quite…

- While that will work, Ansible has modules for apt, so why not use that instead

```
ansible webservers -s -m apt -a 'pkg=nginx state=installed
update_cache=true'
```

Software removal as well

- We use almost the same command

```
ansible webservers -s -m apt -a 'pkg=nginx
state=absent update_cache=true'
```

Ansible one-liners

- So far we have seen one-liners as ad-hoc commands
- You can accomplish much with them, but when you formalize them and add some structure, you get much more
- You get playbooks

Ansible Playbooks

- This is where the complexity starts
  - What if we wanted nginx to be started?
  - How about if we need to make sure nginx was installed and then started if it was available?
- Playbooks are the answer
  - I am not going to get into complicated playbooks here, it is a very detailed thing once you start using Ansible

Example Playbook

- A very basic playbook

```
---
- hosts: webservers
  tasks:
    - name: Install Nginx
      apt: pkg=nginx state=installed \
      update_cache=true
```

Playbook run output

```
network@ansible:~$ ansible-playbook -s -K  ./nginx.yaml
SUDO password:

PLAY ***************************************************************************

TASK [setup] *******************************************************************
ok: [node1]
ok: [node4]
ok: [node3]
ok: [node2]

TASK [Install Nginx] ***********************************************************
changed: [node3]
changed: [node2]
changed: [node4]
changed: [node1]

PLAY RECAP *********************************************************************
node1                      : ok=2    changed=1    unreachable=0    failed=0
node2                      : ok=2    changed=1    unreachable=0    failed=0
node3                      : ok=2    changed=1    unreachable=0    failed=0
node4                      : ok=2    changed=1    unreachable=0    failed=0
```

YAML Format

- As you saw, the playbook format is YAML[1]
  - It is easy to write
  - Easy to read
  - Allows for lots of comments, so pretty much self-documenting
  - Simple text file

[1] YAML Ain't Markup Language

Back to complexity

- Playbooks are where you start to get into more formal declarations for your servers
- This is the beginning of a configuration management system
- From here you add roles and can include other files
  - We are not covering that tonight, you can read up on it if you are interested

Modules

- You may have noticed the –m in the commands
- That tells ansible what module to use for the command
  - The first command used the ping module
  - The second one used the shell module
  - There are many more

Modules come in two forms

- Core modules
  - These are modules that the core ansible team maintains and will always ship with ansible itself. They will also receive slightly higher priority for all requests than those in the “extras” repos.
- Extras modules
  - These modules are currently shipped with Ansible, but might be shipped separately in the future. They are also mostly maintained by the community. Non-core modules are still fully usable, but may receive slightly lower response rates for issues and pull requests.

Types of Module

- Cloud Modules
- Clustering Modules
- Commands Modules
- Database Modules
- Files Modules
- Inventory Modules
- Messaging Modules
- Monitoring Modules
- Network Modules
- Notification Modules
- Packaging Modules
- Source Control Modules
- System Modules
- Utilities Modules
- Web Infrastructure Modules
- Windows Modules

Files Modules

- acl - Sets and retrieves file ACL information.
- assemble - Assembles a configuration file from fragments
- blockinfile (E) - Insert/update/remove a text block surrounded by marker lines.
- copy - Copies files to remote locations.
- fetch - Fetches a file from remote nodes
- file - Sets attributes of files
- find - return a list of files based on specific criteria
- ini_file - Tweak settings in INI files

Files Modules

- lineinfile - Ensure a particular line is in a file, or replace an existing line using a back-referenced regular expression.
- patch (E) - Apply patch files using the GNU patch tool.
- replace - Replace all instances of a particular string in a file using a back-referenced regular expression.
- stat - retrieve file or file system status
- synchronize - Uses rsync to make synchronizing file paths in your playbooks quick and easy.
- template - Templates a file out to a remote server.
- unarchive - Unpacks an archive after (optionally) copying it from the local machine.
- xattr - set/retrieve extended attributes

Example of “lineinfile”

```
- name: update ssh parameters
  lineinfile:
    dest=/etc/ssh/sshd_config
    state=present
    regexp=^{{ item.key }}
    line="{{ item.key }} {{ item.value }}"
    insertafter=EOF
  with_items:
    - { key: 'PermitRootLogin', value: 'no' }
    - { key: 'LoginGraceTime', value: '20' }
    - { key: 'X11Forwarding', value: 'no' }
    - { key: 'ClientAliveInterval', value: '30' }
    - { key: 'ClientAliveCountMax', value: '1000' }
  notify:
    - restart ssh
```

Using VirtualBox for the demo

DEMO

Why is it called ansible?

- An ansible is a fictional machine capable of instantaneous or superluminal communication. It can send and receive messages to and from a corresponding device over any distance whatsoever with no delay. Ansibles occur as plot devices in science fiction literature.

Source: Wikipedia

RedHat acquires Ansible

RALEIGH, N.C. — October 16, 2015 —

Red Hat, Inc. (NYSE: RHT), the world's leading provider of open source solutions, today announced that it has signed a definitive agreement to acquire Ansible, Inc., a provider of powerful IT automation solutions designed to help enterprises move toward frictionless IT. Ansible's automation capabilities, together with Red Hat's existing management portfolio, will help users drive down the cost and complexity of deploying and managing both cloud-native and traditional applications across hybrid cloud environments. With today's announcement, Red Hat expands its leadership in hybrid cloud management, OpenStack and containers.

Future Talk

- Assuming this was of interest, I'm planning a more detailed talk that walks through the entire creation of a small cluster of computers:
  - Initial configuration
  - Provisioning software
  - Updating
  - Reporting

Resources

- http://www.ansible.com/
- http://docs.ansible.com/
- http://www.mechanicalfish.net/start-learning-ansible-with-one-line-and-no-files/
- https://wiredcraft.com/blog/getting-started-with-ansible-in-5-minutes/
- http://blog.dhananjaynene.com/2013/10/hands-on-ansible-tutorial/
- The Bugs Bunny graphic is from the Hansel & Gretel episode where characters keep repeating “Hansel?” all the time. Bugs is still under copyright, currently Warner Home Video