Configuration management with Ansible and Git

Paul Waring (paul@xk7.net, @pwaring)

January 10, 2017

Topics

- Configuration management
- Version control
- Firewall
- Apache / nginx
- Git Hooks
- Bringing it all together

Configuration management

- Old days: edit files on each server, manual package installation
- Boring, repetitive, error-prone
- Computers are good at this sort of thing
- Automate using shellscripts - doesn't scale, brittle
- Create configuration file and let software do the rest
- Less firefighting, more tea-drinking

Terminology

- Managed node: Machines (physical/virtual) managed by Ansible
- Control machine: Runs the Ansible client
- Playbook/manifest: Describes how a managed node is configured
- Agent: Runs on managed nodes

Ansible

- One of several options
- Free and open source software - GPLv3
- Developed by the community and Ansible Inc.
- Ansible Inc now part of RedHat
- Support via the usual channels, free and paid

Alternatives to Ansible

- CfEngine
- Puppet, Chef
- SaltStack

Why Ansible?

- Minimal dependencies: SSH and Python 2
- Many Linux distros ship with both
- No agents/daemons (except SSH)
- Supports *really* old versions of Python (2.4 / RHEL 5) on nodes
- Linux, *BSD, macOS and Windows

Why Ansible?

- Scales up and down
- Gentle learning curve
- But… no killer features
- A bit like: vim vs emacs

Configuration file

- Global options which apply to all nodes
- INI format
- Write once, then leave

Configuration file

```
[defaults]
inventory = hosts
```

Inventory file

- List of managed nodes
- Allows overriding of global options on per-node basis
- Group similar nodes, e.g. web servers

Inventory file

```
[staging]
vagrant ansible_host=127.0.0.1
  ansible_port=2222
  ansible_user=vagrant
  ansible_private_key_file=
    ~/.vagrant.d/insecure_private_key

[production]
bigv ansible_host=bigv.ukuug.org
  ansible_user=root
```

Modules

- Abstraction of functionality, e.g. create accounts
- Core, Extras and Third Party
- Mostly Python, can use other languages too

Playbooks

- List of tasks to run on nodes
- Imperative vs declarative
- Can (and should!) be idempotent
- Yet Another Markup Language (YAML)

Firewall playbook

```yaml
- name: Security playbook
  hosts: vagrant
  become: yes
  become_user: root

  tasks:
    - name: enable incoming ssh
      ufw:
        rule: allow
        to_port: ssh
```

Firewall playbook

```
- name: allow all outgoing traffic
  ufw:
    direction: outgoing
    policy: allow

- name: deny all incoming traffic
  ufw:
    direction: incoming
    policy: deny
    log: yes
```

Web playbook

```
vars:
  install_packages:
    - apache2
    - libapache2-mod-php5
    - php5-mysql
  apt_update_cache: yes

tasks:
  - name: Install Apache and related packages
    with_items: "{{ install_packages }}"
    apt:
      name: "{{ item }}"
      update_cache: "{{ apt_update_cache }}"
      cache_valid_time: 3600
```

Web playbook

```yaml
- name: Start Apache
  service:
    name: apache2
    state: started
```

Handlers

```
- name: enable vhost configuration files
      with_items: vhosts_files
      file:
        src: "{{ vhosts_available_dir }}/{{ item }}"
        dest: "{{ vhosts_enabled_dir }}/{{ item }}"
        state: link
      notify: reload apache

handlers:
  - name: reload apache
    service: name=apache2 state=reloaded
```

Git

- Written for Linux kernel development
- Distributed - each copy is a repository
- Alternatives: Mercurial (Mozilla), GNU Bazaar (Ubuntu)
- Git has won the DVCS wars

Git features

- Rollback/undo changes, e.g. `git checkout -- <file>`
- View full history to the beginning of time: `git log`
- Branching is cheap

Git workflow

- Create branch: `git branch <branch>`
- Checkout branch: `git checkout <branch>`
- Add files: `git add <file>`
- Commit changes: `git commit -m <message>`
- Merge: `git checkout master && git merge <branch>`
- Push: `git push`

Git hooks

- Perform actions at given points in workflow
- Example: *pre-commit* (unit tests)
- Example: *post-commit* (deployment)

Pre-commit

```
#!/bin/bash

files=$(git diff --staged --name-only --diff-filter=MA \
 | grep -E "ansible/[^/]*\.yml")

for filepath in $files; do
  ansible-playbook --syntax-check $filepath -i localhost
  status=$?

  if [ $status != 0 ]; then
    echo "Syntax check failed on: ${filepath}"
    exit $status
  fi
done

exit 0
```

Post-commit

```bash
#!/bin/bash

export ANSIBLE_CONFIG="${PWD}/ansible/ansible.cfg"
export HOSTS_FILE="${PWD}/ansible/hosts"

files=$(git log --name-only --pretty=format: \
  --diff-filter=MA -n 1 \
 | grep -E "ansible/[^/]*\.yml")

for filepath in $files; do
  ansible-playbook ${filepath} -i ${HOSTS_FILE}
done
```

Questions?

- Slides at:
  `https://github.com/pwaring/technical-talks`