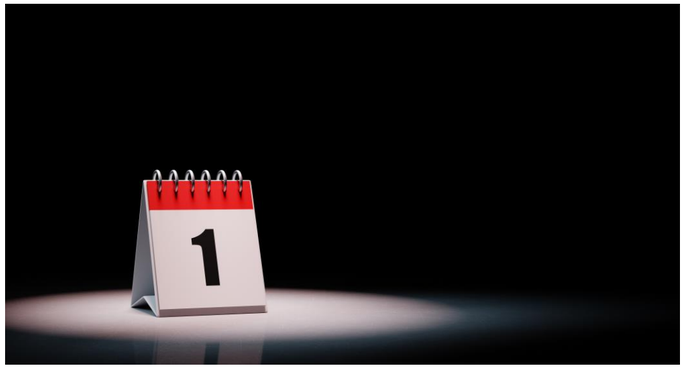
Working with the Big Three

Andrew Mallett
Linux Author and Trainer

@theurbanpenguin www.theurbanpenguin.com

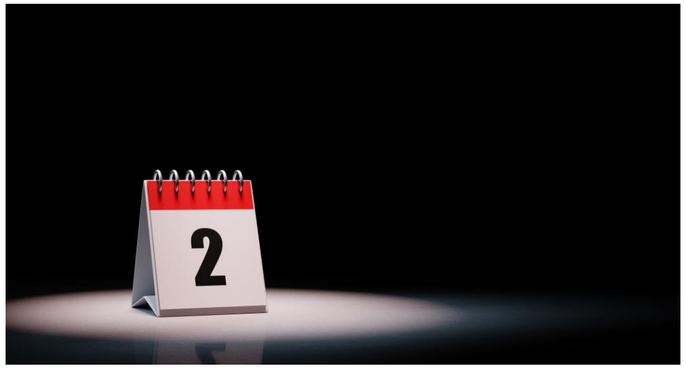
The Big Three

When working with computer OSs we have three main elements we manage all the time



Packages

Installing software



Files

Adding and editing files

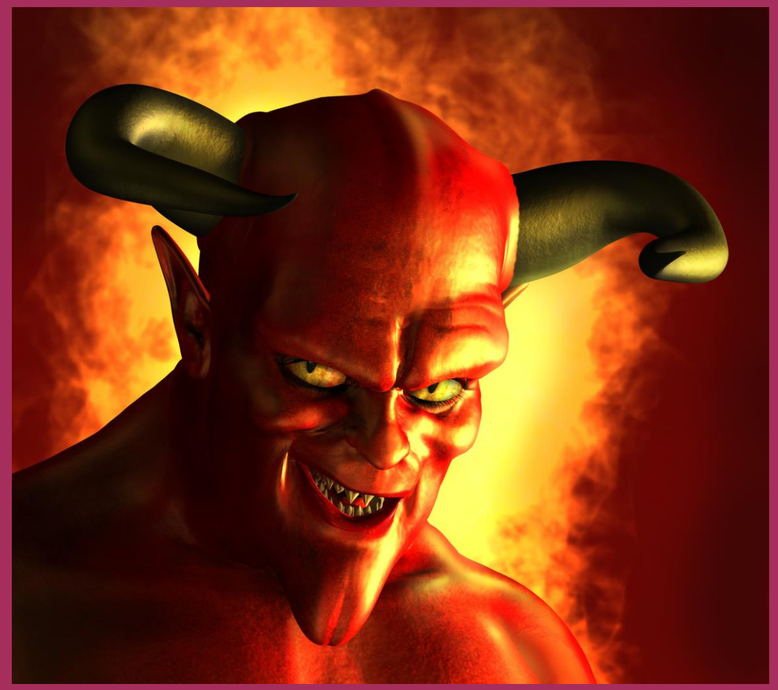
Services

Managing services

Overview

Working with the big 3

- Deploy Apache
  - The apache webserver package
  - Copy web page
  - Start web service
  - Variables to cater for differences
- Manage Chronyd
  - Ensure chrony deployed
  - Deploy standardized configuration
  - Restart service is configuration changed



Agnostic

While Ansible tries to be agnostic, there are still areas where an OS can trip up any configuration management system. Mainly where package and service names differ

RedHat vs Debian

RedHat

For the apache web server, we use httpd for both package and service names

Debian

For the apache web server, we use apache2 for both package and service names

```
- name: 'Manage Apache Deployment'
  hosts: Redhat
  become: true
  gather_facts: false
  tasks:
    - name 'Install Apache Web Server'
      package:
        name: 'httpd'
        state: 'present'
```

Installing Apache

Installing Apache will work without issues across the RHEL and CentOS systems but, hard coding the package to httpd will fail on the Ubuntu system as it uses the package name apache2

```
- name: 'Add welcome page'
      copy:
        dest: '/var/www/html/index.html'
        state: 'file'
        content: '<h1>Welcome to the web site</h1>'
```

Adding Web Content

The web content is simpler as the path is consistent on RedHat based systems and Debian based systems. The copy module has a lot of flexibility to investigate that add interest

```
- name: 'Start and enable web server'
      service:
        name: 'httpd'
        state: 'started'
        enabled: true
```

Managing the Service

We are back to having issues where the service name differs

Demo

Inventory Review

- Listing Inventory
- Listing Membership by Host

Demo

Managing Apache on RedHat

- install httpd
- start and enable service

Demo

Managing Apache on RedHat

- Using copy module
  - with content
  - with src
  - copy directories

Helping Ansible

We can help Ansible by using logic or variables within the Playbook, creating groups variables we can correctly make the stetting across all three systems

```
$ mkdir ~/group_vars

$ vim ~/group_vars/Redhat
apache_pkg: httpd
apache_srv: httpd

$ vim ~/group_vars/ubuntu
apache_pkg: apache2
apache_svc: apache2
```

Using Ansible's Group Variables

The inventory file we have created includes the Redhat group which includes the rhel and stream groups. The ubuntu group is for the single ubuntu system

```
- name: 'Manage Apache Deployment'
  hosts: all
  become: true
  gather_facts: false
  tasks:
    - name 'Install Apache Web Server'
      package:
        name: "{{ apache_pkg }}"
        state: 'present'
```

Installing Apache on All Hosts

Using the variables, we are able to easily able to manage Apache across all hosts.

Demo

Adding Variables

- Create group variables
- Modify Playbook to use variables

```
$ wc -l /etc/chrony.conf
38 /etc/chrony.conf

$ grep -Ev '^($|#)' /etc/chrony.conf | wc -l
7

$ mkdir ~/ansible/chrony ; grep -Ev '^($|#)' /etc/chrony.conf >
~/ansible/chrony/chrony.conf
```

Configuring Chrony the Time Service

On both Ubuntu and RedHat the configuration for Chrony is we documented. The same file will work across all systems allowing us to clean the file and use this as the corporate time configuration file

```
$ echo 'chrony_conf: /etc/chrony.conf' >>  ~/group_vars/Redhat
$ echo 'chrony_svc: chronyd' >>  ~/group_vars/Redhat

$ echo 'chrony_conf: /etc/chrony/chrony.conf' >> ~/group_vars/ubuntu
$ echo 'chrony_svc: chrony' >> ~/group_vars/ubuntu
```

Chrony Variables

The path of the configuration file varies between the system, as does the service name

Handlers

Ansible tasks are executed on each Play run, handlers only execute if notified. We can use this to only restart a service if the configuration file changes

```
tasks:
  - name: 'Manage Chrony Configuration'
    copy:
      src: 'chrony.conf'
      dest: "{{ chrony_conf }}"
    notify: restart_chrony

handlers:
  - name: 'restart_chrony'
    service:
      name: "{{ chrony_svc }}"
      state: 'restarted'
```

Implementing Handlers

Handlers only execute when notified. We can use the notify meta-parameter in the copy module to restart Chrony is the configuration changes

Demo

Managing Chrony

- Clean configuration file
- Adding variables

Demo

Managing Chrony

- Creating the Chrony Playbook

Summary

Configuring the Big Three

- package
- service
- copy
  - Apache
    - Variables for service and package
    - Copy directory to document root
  - Chrony
    - Variables for configuration path
    - Clean configuration with grep
    - Handler to restart service

Managing Users in Ansible