Configuration Management

Linux System Administration
Spring 2026

How to manage multiple machines

- Scripts that configure users and services
  - Very flexible, can accommodate heterogeneity
  - Ad hoc procedures can be confusing
- Configuration Management tools
  - Declarative desired state of each machine
  - Knows common tasks like user & network configuration
  - Manage large number of machines

CM goals

- idempotent
  - Applying multiple times does not create duplicates
- cross platform
  - Works across flavors of Linux or Unix
    - Some also support Windows
- distributed
  - Apply to all managed machines

Commonly used CM systems

System	Initial Release	Implementation	Configuration	Web site
CFEngine	1993	C	custom	cfengine.com
Puppet	2005	Ruby	custom	puppet.com
Chef	2009	Ruby	Ruby	chef.io
Salt	2011	Python	YAML	saltstack.com
Ansible	2012	Python	YAML	ansible.com

and many more...

https://en.wikipedia.org/wiki/Comparison_of_open-source_configuration_management_software

Using a CM

- Pros
  - Consistent configuration on all hosts
  - Declarative syntax
  - Cross platform
  - Remote management
  - Easier to document
- Cons
  - Errors can take down many hosts
  - CM differ in approach and terminology
  - Manual config/CM conflicts makes ad hoc changes difficult

CM User Example

- Puppet

```
user { "mscott":
    uid => "2001",
    ensure => present,
    comment => "Michael Scott",
    gid => "mscott",
    groups => ["managers"],
    shell => "/bin/bash",
    home => "/home/mscott",
    managehome => true,
}
```

- Ansible

```
name: User Michael Scott
ansible.builtin.user:
   name: mscott
   uid: 2001
   state: present
   comment:  "Michael Scott"
   group: mscott
   groups: managers
   shell: /bin/bash
   home: /home/mscott
   create_home: yes
```

Installing Puppet

- Install package on all machines
  - `dnf install puppet`
  - `apt install puppet`
- Install modules
  - `puppet module install puppet-network`
- Apply a puppet manifest
  - `puppet apply dmusers.pp`

Puppet Modules

- Public repo `https://forge.puppet.com`
  - Quality and age varies by module
- Builtin modules for basic things
  - files and directories
  - users & groups
- Puppet can be run as client/server
  - Requires setting up puppetserver and certs
  - Manifests often just copied and then applied

Ansible Concepts

- Control node used to manage network
- Managed nodes
  - Accessed via ssh
  - Does not have ansible installed
- Module (code to accomplish tasks)
- Playbook (things to accomplish)
  - YAML file
  - Task (what we want to accomplish)

Installing Ansible

- Ansible need only be installed on the control node

  `dnf -y install ansible` or `apt -y install ansible`

- Access to managed hosts is by ssh
  - Set yourself up for password-less logins
- Configure ansible
  - `/etc/ansible/ansible.cfg` add `pipelining=true`
  - `/etc/ansible/hosts` configure `saclass` hosts

Applying ansible

- Show hosts in saclass group
  - `ansible saclass --list-hosts`
- Check connectivity
  - `ansible saclass -m ping`
- Running privileged commands
  - `ansible-playbook -K umask.yaml`
  - `-K` prompt for the `BECOME` password

Play settings

`name`: Name of the play

`hosts`: Hosts to apply this play to

`remote_user`: The ssh user on the managed node

`become`: Run command on managed node as another

`become_user`: User to run remote command as

`tasks:` What to do

Running a play

```
vlakkies@machinee $ ansible-playbook -K umask.yaml
BECOME password:

PLAY [default umask]
*********************************************************************************************************************

TASK [Gathering Facts]
*********************************************************************************************************************
ok: [100.64.41.6]
ok: [100.64.41.3]
ok: [100.64.41.2]
ok: [100.64.41.4]
ok: [100.64.41.1]

TASK [Set /etc/profile.d/umask.sh] ********************************************************************************
changed: [100.64.41.3]
changed: [100.64.41.6]
changed: [100.64.41.1]
changed: [100.64.41.2]
changed: [100.64.41.4]

PLAY RECAP *********************************************************************************************************
100.64.41.1                : ok=2    changed=1    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
100.64.41.2                : ok=2    changed=1    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
100.64.41.3                : ok=2    changed=1    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
100.64.41.4                : ok=2    changed=1    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
100.64.41.6                : ok=2    changed=1    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
```

Security considerations

- Relies on ssh to access remote machines
  - Requires one credential for access
  - Requires one credential for sudo access
  - An action should require ONE password
- Small site: Use sysadmin credentials
- Large site: Some reasonable compromise

Ansible builtin modules

- `ping` verify basic connectivity
- `user` manage user accounts
- `group` manage user groups
- `file` manage files and propeties
- `copy` copy files
- `template` copy a file based on a template
- `systemd` manage systemd services
- `cron` manage crontab and cron.d entries
- `dnf` manage packages using dnf
- `apt` manage packages using apt