Cross-Distro Ansible Roles

Shawn T. Amundson
Bitwise IO, Inc.

Who Am I?

- Long-time Linux user, sysadmin, developer
- A (fairly new) user of Ansible
- Co-Founder of Bitwise IO, Inc.

A Bit of Theory

- A single way to configure a service across different Linux distributions is Good.
- Making a complex system maintainable requires breaking it down into building blocks.
- Ansible roles can be those building blocks.
- Building blocks should be simple, predictable, and without side effects.

Approach

- Add support for one distribution at a time
- Avoid skipped tasks when possible

Example: NFS Configuration

- NFS server configuration is essentially the same across all distributions
- Need to install packages (via yum or apt)
- Need to enable and start services (rpcbind, nfs, nfslock, etc.)
- Need to configure /etc/exports
- Need to configure NFS settings (in particular, ports used by NFS components - mountd, statd, lockd, etc.)
- Need to configure the system's firewall

Goals

- Create nfs-server role
- Work with “minimal” distribution install
- Support CentOS 6, CentOS 7, Ubuntu 14.04
- Nothing else required to setup a NFS server!

CentOS 6

- Files:

  ```
  /etc/exports
  /etc/sysconfig/nfs
  ```

- Install packages (via yum): libselinux-python, nfs-utils, rpcbind
- Services: rpcbind, nfs, nfslock
- Configure iptables

template vs. lineinfile

- For /etc/exports, the role uses template because:
  - the role is the “source of truth” for the entire file
  - removing export lines is elegantly handled with a template
- For /etc/sysconfig/nfs, the role uses lineinfile because:
  - sysadmin may manually edit /etc/sysconfig/nfs to modify other settings (or via another role…)

Initial Files

```
hosts
host_vars/train
nfs-servers.yml
roles/nfs-server/handlers/main.yml
roles/nfs-server/tasks/iptables.yml
roles/nfs-server/tasks/main.yml
roles/nfs-server/templates/exports.j2
roles/nfs-server/vars/RedHat-6.yml
```

hosts:

```
[nfs-servers]
thrain
```

host_vars/thain:

```
---
nfs_server:
  exports:
    - { directory: /tmp, options: "10.9.0.0/24(ro)" }
```

nfs-servers.yml:

```
---
- hosts: nfs-servers
  remote_user: root
  roles:
    - role: nfs-server
```

Run: `ansible-playbook -i hosts -l thain nfs-servers.yml` roles/nfs-server/templates/exports.j2:

```
{% for export in nfs_server.exports %}
{{ '%-20s' % export.directory }}
{{ export.options }}
{% endfor %}
```

Example result:

```
/tmp                 10.9.0.0/24(ro)
```

roles/nfs-server/vars/RedHat-6.yml:

```
---

nfs_server_packages:
  - libselinux-python
  - nfs-utils
  - rpcbind

nfs_server_services:
  - rpcbind
  - nfs
  - nfslock

nfs_server_config_settings:
  - { file: '/etc/sysconfig/nfs', key: 'MOUNTD_PORT', value: 892 }
  - { file: '/etc/sysconfig/nfs', key: 'STATD_PORT', value: 662 }
  - { file: '/etc/sysconfig/nfs', key: 'LOCKD_TCPPORT', value: 32803 }
  - { file: '/etc/sysconfig/nfs', key: 'LOCKD_UDPPORT', value: 32769 }

nfs_server_firewall_ports:
  - { port: 111, proto: 'tcp' }
  - { port: 111, proto: 'udp' }
  - { port: 662, proto: 'tcp' }
  - { port: 662, proto: 'udp' }
  - { port: 892, proto: 'tcp' }
  - { port: 892, proto: 'udp' }
  - { port: 2049, proto: 'tcp' }
  - { port: 2049, proto: 'udp' }
  - { port: 32803, proto: 'tcp' }
  - { port: 32769, proto: 'udp' }
```

roles/nfs-server/tasks/main.yml:

```
---
- name: include distribution specific variables
  include_vars: "{{ item }}"
  with_first_found:
    - "{{ ansible_distribution }}-{{ ansible_distribution_major_version }}.yml"
    - "{{ ansible_os_family }}-{{ ansible_distribution_major_version }}.yml"

- name: ensure NFS packages are installed
  yum: pkg={{ item }} state=present
  with_items: nfs_server_packages

- name: ensure /etc/exports is configured
  template: src=exports.j2 dest=/etc/exports owner=root group=root mode=0755
  notify: re-export directories

- name: ensure NFS settings are configured
  lineinfile: dest="{{ item.file }}"
              regexp="^#?{{ item.key }}\s*="
              line="{{ item.key }}={{ item.value }}"
  with_items: nfs_server_config_settings
  notify: restart nfs services

- name: ensure NFS services are enabled and started
  service: name={{ item }} enabled=yes
  with_items: nfs_server_services
  notify: restart nfs services

- include: iptables.yml
```

roles/nfs-server/tasks/iptables.yml:

```
---

- name: get current iptables NFS rules
  shell: /sbin/iptables -S NFS || /bin/true
  register: iptables_nfs_rules

- name: get current iptables NFS rules
  command: /sbin/iptables -S
  register: iptables_all_rules

- name: ensure iptables NFS chain exists
  command: /sbin/iptables -N NFS
  when: iptables_nfs_rules.stdout.find("-N NFS") == -1
  notify: save iptables

- name: ensure iptables NFS chain is used
  command: /sbin/iptables -I INPUT 1 -j NFS
  when: iptables_all_rules.stdout.find("-j NFS") == -1
  notify: save iptables

- name: ensure fireall ports are open
  command: /sbin/iptables -A NFS -p {{ item.proto }}  --dport {{ item.port }} -j
ACCEPT
  when: iptables_nfs_rules.stdout.find(" -m {{ item.proto }} --dport
{{ item.port }} ") == -1
  with_items: nfs_server_firewall_ports
  notify: save iptables
```

- Result of iptables tasks:

```
[root@thrain ~]# iptables -S NFS
-N NFS
-A NFS -p tcp -m tcp --dport 111 -j ACCEPT
-A NFS -p udp -m udp --dport 111 -j ACCEPT
-A NFS -p tcp -m tcp --dport 662 -j ACCEPT
-A NFS -p udp -m udp --dport 662 -j ACCEPT
-A NFS -p tcp -m tcp --dport 892 -j ACCEPT
-A NFS -p udp -m udp --dport 892 -j ACCEPT
-A NFS -p tcp -m tcp --dport 2049 -j ACCEPT
-A NFS -p udp -m udp --dport 2049 -j ACCEPT
-A NFS -p tcp -m tcp --dport 32803 -j ACCEPT
-A NFS -p udp -m udp --dport 32769 -j ACCEPT
```

Works great, but…

- We want flexibility, but what if the user wanted to configure iptables via another role?
- The obvious thing to do is… make iptables tasks an optional part of the role via a host_vars setting, but…
    - we could use “when” on every iptables task, but it will print “skipped” for each task which might be confusing to the user (and we want predictability)
    - we could use “when” on the include of iptables.yml, but it’s the same thing as per-task
- Problems like this often indicate should split the role up

“Elegant” solution?

- We can separate the iptables into it’s own role “nfs-server-iptables”
- The new role can be included in the playbook appropriately
- We really want it “off by default”, so requiring a user to expressly include it seems okay.
- Disadvantage: sharing an additional role with others will be much harder than the boolean config value would have been
- Still, seems like the best option

- Files after splitting into two roles:

```
hosts
host_vars/thrain
nfs-servers.yml

roles/nfs-server/handlers/main.yml
roles/nfs-server/tasks/main.yml
roles/nfs-server/templates/exports.j2
roles/nfs-server/vars/RedHat-6.yml

roles/nfs-server-iptables/handlers/main.yml
roles/nfs-server-iptables/tasks/main.yml
roles/nfs-server-iptables/vars/RedHat-6.yml
```

- Content is the same…

Support for CentOS 7

- Add new vars file: roles/nfs-server/vars/RedHat-7.yml

```
---

nfs_server_packages:
  - nfs-utils

nfs_server_services:
  - rpcbind
  - nfs-server
  - nfs-lock

nfs_server_config_settings:
  - { key: 'LOCKD_TCPPORT', value: 32803 }
  - { key: 'LOCKD_UDPPORT', value: 32769 }
```

- Add new role: nfs-server-firewalld

Support for Ubuntu 14.04

- Add new file: roles/nfs-server/vars/Ubuntu-14.yml

```
---

nfs_server_packages:
  - nfs-kernel-server

nfs_server_services:
  - nfs-kernel-server
  - statd

nfs_server_config_settings:
  - { file: '/etc/default/nfs-kernel-server', key: 'RPCMOUNTDOPTS',
value: "\\\"--manage-gids -p 20048\\\"" }
  - { file: '/etc/default/nfs-common', key: 'STATDOPTS', value: "\\
\"--port 662\\\"" }
```

- Changes to package management:

```
- name: ensure NFS packages are installed (yum)
  yum: pkg={{ item }} state=present
  with_items: nfs_server_packages
  when: ansible_pkg_mgr == "yum"

- name: ensure NFS packages are installed (apt)
  apt: name={{ item }} state=present
  with_items: nfs_server_packages
  when: ansible_pkg_mgr == “apt"
```

- Additional Ubuntu-specific task:

```
- name: ensure NFS settings are configured (modprobe.d/options.conf)
  lineinfile: dest=/etc/modprobe.d/options.conf
              regexp="options lockd nlm"
              line="options lockd nlm_udpport=32769 nlm_tcpport=32803"
              create=yes
  notify: restart nfs services
  when: ansible_os_family == "Debian"
```

Summary

- Support for CentOS 6, CentOS 7, Ubuntu 14
- Three roles (building blocks):
  nfs-server, nfs-server-iptables, nfs-server-firewalls
- Very few “skipped tasks”

Development Environment

- KVM
- One VM per supported distribution
- Minimal installs
- “Snapshots” for re-testing (cp of images)

NFS Testing

- On the NFS server:

```
rpcinfo -p
```

- On the NFS client:

```
rpcinfo -u <hostname> status
mount <hostname>:/tmp /mnt
```

Issues

- This example isn’t quite done:
  - CentOS 7 - broken until nfs-server is restarted or upon a reboot
  - CentOS 7 - "rpcinfo -u balin status" broken (statd port not set)
  - Ubuntu 14 - requires a reboot for lockd module to reload

Potential Future Improvements

- Specify ports via host_vars
- Tune more NFS settings
- Create directories automatically
- More distributions: SUSE, etc.
- nfs-server-ufw

- Presentation/roles available on github:

https://github.com/bitwiseio/presentations-ansible-nfs-server