Ansible

Michael Pobega (pobegam@sunyit.edu)

May 30, 2017

Abstract

Documentation and disambiguation of Ansible, ansible playbooks and configuration files.

Contents

1 Quick System Information

1.1 Software Versions

- **Oracle Enterprise Linux** release 6.5 (Santiago)
- **Kernel** 3.8.13-26.1.1.el6uek.x86_64
- **Ansible** 1.4.3

1.2 Important File Locations

- **Ansible host configuration file** /etc/ansible/hosts
- **Ansible playbooks** ~ansible/scripts/
- **Playbook file storage** /srv/nfs

2 Ansible

2.1 Installation

Ansible is installed from the EPEL repositories. Ansible uses SSH to connect to hosts and therefore doesn't require any additional client software.

2.2 Ansible Host Setup

2.2.1 Installation

Enable EPEL and install the *ansible* package. This should pull in ansible and all dependencies required to use it.

2.2.2 Configuration

All configuration is done in the /etc/ansible directory.
/etc/ansible/ansible.cfg is ansible's configuration file. The defaults should be fine.
/etc/ansible/hosts is the host file. This file lists all of the hosts Ansible is aware of. The hosts are organized into groups. One host can be in multiple groups.

2.3 Ansible Client Setup

Setting up client nodes for Ansible to connect to is a three step process.

1. Add the *ansible* user to the system. Give it a secure password. This password can be a throwaway, as we are going to be using SSH keys to login.
2. Use ssh-copy-id to transfer the Ansible account's SSH key to the new client.
3. Give ansible full access through sudo. Passwordless is preferred for simplicity, but is a security problem.

2.4 Using Ansible

To use Ansible, ssh to avl-sysans-001 and su into the ansible user. Run *ssh-agent bash && ssh-add* to enable passwordless logins for the ansible account. Now you can invoke Ansible on the command line!

3 Playbooks

3.1 nagios/nrpe.yml

This is the Nagios NRPE playbook. When this playbook is run against a group of hosts (specified on the command line individually using -i, or with –extra-hosts="hosts=foo") it installs and configures the NRPE daemon on the host. It currently **does not** check if NRPE is already installed, so be warned if you have installed NRPE from EPEL this will install a separate version of NRPE which may conflict.

This playbook compiles NRPE locally and uses xinetd to start/stop the daemon. It also pulls NRPE from an NFS server so that the version installed can be monitored and controlled - this behavior can be changed but will lead to different servers having wildly different versions of NRPE installed, and down the line this will cause headaches. I recommend trying to keep every server on the same version of NRPE.

3.2 nagios/custom-commands.yml

This playbook controls the custom-commands.cfg file on all of the remote NRPE hosts. The first thing it does is check to see if NRPE is installed and if so how it is installed and sets the appropriate variables for each host.

```
- name: check if /etc/nrpe.d/ exists
  stat: path=/etc/nrpe.d/
  register: nrpe_exists
- name: check if /etc/init.d/nrpe exists
  stat: path=/etc/init.d/nrpe
  register: nrpe_initd
- name: check if /etc/xinetd.d/nrpe exists
  stat: path=/etc/xinetd.d/nrpe
  register: nrpe_xinetd
```

custom-commands.yml: figuring out how nrpe is installed

Next it uploads the custom-commands.cfg file from the Ansible server to the remote servers and chowns them based on how NRPE is installed on that system

```
- name: Uploading $REMOTE$:/etc/nrpe.d/custom-commands.cfg for yum
    install
  copy: src=/srv/nfs/nrpe/custom-commands.cfg
      dest=/etc/nrpe.d/custom-commands.cfg owner=nrpe group=nrpe
      mode=750 force=yes
  when: nrpe_exists.stat.exists and nrpe_initd.stat.exists
```

```
  - name: Uploading $REMOTE$:/etc/nrpe.d/custom-commands.cfg for src
      install
    copy: src=/srv/nfs/nrpe/custom-commands.cfg
        dest=/etc/nrpe.d/custom-commands.cfg owner=nagios group=nagios
        mode=750 force=yes
    when: nrpe_exists.stat.exists and nrpe_xinetd.stat.exists
    [...]
```

custom-commands.yml: uploading and chowning custom-commands.cfg

This script also uses sed to swap the path to the Nagios plugin directory based on how NRPE is installed using similar metrics

```
  - name: Updating $REMOTE$:/etc/nrpe.d/custom-commands.cfg for yum
      install
    command: sed -i "s#\[NAGPLUGDIR\]#/usr/lib64/nagios/plugins#g"
        /etc/nrpe.d/custom-commands.cfg
    when: nrpe_exists.stat.exists and nrpe_initd.stat.exists
    [...]
```

custom-commands.yml: pointing to the right Nagios plugin directory

After this it copies over all of the files in /srv/nfs/nrpe/libexec (local to the Ansible server) to the remote server's plugin/libexec directory, and restarts the daemon so that the changes take effect.

One advantage of this setup is that you **can** compile nrpe with arguments/-commands disabled, which increases security slightly (official documentation claims enabling this feature may lead to security hazards in an unencrypted environment), but arguments are left on by default for these installs for legacy compatibility.

3.3 servicedesk.yml

This playbook sets up the Service Desk script and cronjob for logging. The playbook is very basic:

```
- hosts: $hosts
  sudo: yes
  tasks:

  - name: Create scripts directory if it doesnt exist
    shell: mkdir -p /usr/local/adm/scripts/

  - name: copy ServiceDesk.sh to remote host
    copy: owner=root group=root mode=640 src=/srv/nfs/ServiceDesk.sh
        dest=/usr/local/adm/scripts/ServiceDesk.sh

  - name: add cronjob
    cron: cron_file=../crontab name=ServiceDesk minute=0 hour=15
        user=root job=/usr/local/adm/scripts/ServiceDesk.sh
```

This script needs to be run with the -i or –extra-hosts=, as it is meant to be run on machines that don't have the ServiceDesk.sh script setup already (though rerunning it on a machine that already has it doesn't have any adverse effects). It then sets a cronjob that runs every day at 3 PM as the user root.

Pretty simple playbook. This is now part of the default template for new VMs that are created, so this should only have to be run on legacy systems that don't have ServiceDesk installed.

3.4 dnssec.yml

Playbook used to deploy the sunyct.edu DNSSEC servers. It installs openssl[-devel], rng-tools and keepalived. Then it configures rng-tools to use /dev/urandom instead of /dev/random.

```
- name: put EXTRAOPTIONS="r /dev/urandom" in /etc/sysconfig/rngd
  shell: sed -i s#EXTRAOPTIONS=.*#EXTRAOPTIONS='"'"-r
      /dev/urandom"'"'#g /etc/sysconfig/rngd
```

dnssec.yml: /dev/urandom replacement

It fetches and compiles bind9.9, puts the sunyct.edu keys in the /etc/bind/keys directory and configures keepalived.

This playbook makes use of the host variable features of ansible, where you can define variables in the /etc/ansible/hosts file. In this case the variables are used to define the virtual_router_id and virtual_ip of the server, as well as whether the server should be configured as a master or slave.

```
[dnssec]
avl-dnssec-001 role=master virtual_router_id=51 virtual_ip=149.15.3.204
avl-dnssec-002 role=backup virtual_router_id=51 virtual_ip=149.15.3.204
avl-dnssec-003 role=master virtual_router_id=52 virtual_ip=149.15.3.207
avl-dnssec-004 role=backup virtual_router_id=52 virtual_ip=149.15.3.207
```

/etc/ansible/hosts: host variables in the dnssec group

To add more servers as slaves you just add another host with role=backup and run dnssec.yml again.

Note that by default the script will run on every server in the [dnssec] hostgroup, which may or may not be the preferred behavior (but this makes it easier to upgrade bind versions on all hosts simultaneously).

3.5 webblog.yml

This playbook installs and configures httpd and php to use memcache as a session handler on all of the hosts in the [webblg] hostgroup (specified in /etc/ansible/hosts). Was used to deploy the new blog environment (that will run high availability Wordpress).

3.6 syslog.yml

Incomplete. This script will be used to enable remote syslogging to the central rsyslogd server eventually, but for now centralized syslogging is still being researched/investigated.