Software Management: Managing software images with Ansible

Managing software images with Ansible provides a number of advantages when it comes to automation and revision management. Coupling Ansible with a version control system (git for example) may improve the software image workflow and lifecycle.

Best Practice

Best practice for deploying an Ansible managed software image is to clone the default-image generated by the Bright installation and apply your playbooks, tasks and handlers to them based on the role and module.

For example.

Each image may correlate to a type of compute node and be broken down by function. CPU-only or GPU compute node. In Ansible we would consider the role of a system as a compute node with a module that configures it as CPU-only or with a GPU.
It may also have the module that installs the slurm workload manager.

Another role in Ansible might be a storage node with a module that configures Ceph or ZFS.

Step 1: Create a base software image

We will need to build a software image for Ansible to manage. This is done using the standard tools in Bright.

cmsh -c "softwareimage; clone default-image ansible-image; commit"

This will give us an image called ansible-image. We will use this image to install and manage ntp as an example.

Step 2: Install Ansible

Install Ansible inside your image. In this example, we will use the ansible-image we cloned in the previous step.

Please refer to the Ansible documentation for installation methods for your system.

Under RHEL 7 or CentOS 7, Ansible is available in the extras repository.

```
# yum --installroot=/cm/images/ansible-image install ansible
```

Step 3: Setup the playbooks

First, you will need to enter the software image using the chroot command. Some bind mounts are required for ansible to function.

```
# mount -o bind /dev /cm/images/ansible-image/dev

# mount -o bind /proc /cm/images/ansible-image/proc

# mount -o bind /sys /cm/images/ansible-image/sys

# chroot /cm/images/ansible-image
```

Now, once inside the chroot move to the /etc/ansible directory.

Create the site.yml file (/etc/ansible/site.yml)

```
---
# This playbook deploys the whole application stack in this site.
- name: apply common configuration to all nodes
  hosts: all

  roles:
    - common
```

Create a roles directory, with a common directory underneath. In the common folder create a tasks and templates folder. You will also need a group_vars directory under /etc/ansible.

```
# mkdir -p /etc/ansible/roles/common/tasks

# mkdir -p /etc/ansible/roles/common/templates
# mkdir -p /etc/ansible/group_vars
```

Under /etc/ansible/roles/common/tasks create main.yml

```
---
# This playbook contains common plays that will be run on all nodes.

- name: Install ntp
  yum: name=ntp state=present
  tags: ntp

- name: Configure ntp file
  template: src=ntp.conf.j2 dest=/etc/ntp.conf
  tags: ntp

- name: Start the ntp service
  service: name=ntpd enabled=yes
  tags: ntp

- name: test to see if selinux is running
  command: getenforce
  register: sestatus
  changed_when: false
```

Under /etc/ansible/roles/common/templates create ntp.conf.j2

```
driftfile /var/lib/ntp/drift
restrict 127.0.0.1
restrict -6 ::1
server {{ ntpserver }}
includefile /etc/ntp/crypto/pw
keys /etc/ntp/keys
```

Under /etc/ansible/group_vars create all

```
---
# Variables listed here are applicable to all host groups
ntpserver: 192.168.1.1
```

Finally, in /etc/ansible/hosts add an entry for localhost.

Step 4: Running the Ansible Playbook

Navigate to /etc/ansible and run

ansible-playbook -vvv -c local site.yml

Step 5: Applying roles to images based on hostname

As Ansible generally determines the tasks to execute based on the system hostname, there is a utility called chname, which allows a chroot environment to be created with a different hostname. See https://github.com/marineam/chname.

For example, if we wish to have an image for GPU compute nodes, we would create a group in Ansible and apply the role. Update /etc/ansible/hosts with :

[gpu-nodes]
ansible-gpu

Next start the chroot and set the hostname to ansible-gpu.

#chname ansible-gpu chroot /cm/images/ansible-image /bin/bash

Execute Ansible.

cd /etc/ansible

ansible-playbook -c local site.yml

Adding a cpu-nodes group with a node called ansible-cpu in /etc/ansible/hosts would allow us to apply different roles to this image.

This also allows a single Ansible code base which is reusable across all the images.

Caveats

Where possible avoid using Ansible service management tasks to start and stop services. Enabling and disabling services is not an issue. In the chroot environment, Ansible will actually start a service on the real head node if instructed to do so.

Software Management: Managing software images with Ansible

Software Management: Managing software images with Ansible

Unique solution ID: #1396
Author: Simon Brennan
Last update: 2017-10-03 08:12