Configuration Management for the Open Arizona services

1. Goals

1.1. Primary

1.1.1. Move

1.1.1.1. from State Zero (nothing live)

1.1.1.1.1. Github repository

1.1.1.1.2. Ansible control VM

1.1.1.1.3. Service template VM

1.1.1.2. to State One (running service)

1.1.2. With

1.1.2.1. Reliability

1.1.2.2. Repeatability

1.1.2.3. Verifiability

1.2. Secondary

1.2.1. increase level of automation

1.2.2. increase visibility of state

1.2.3. allow for service environments to be near-identical

1.2.4. baseline Ansible as a configuration management solution

2. Implementation of prerequisites

2.1. Created "ualibraries/ansible-config" repository on Github

2.2. Created "ansible1" support virtual machine in LCU1 vSphere

2.2.1. DNS: ansible1.library.arizona.edu <=> 10.130.154.120

2.2.2. CentOS 7 (64-bit)

2.2.3. 2 vCPUs, 4 GB RAM, 20 GB storage

2.2.4. LCU1-Scratch-1 subnet

2.2.5. CIS CentOS 7 benchmark filesystem layout

2.2.6. Minimum install, Standard system security policy

2.2.7. Added:

2.2.7.1. open-vm-tools

2.2.7.2. sysstat

2.2.7.3. ansible (most recent version from EPEL repository)

2.2.7.4. python2-pip

2.2.7.5. pyvmomi (dependency for VMware modules in Ansible)

2.2.7.6. Just for Mike:

2.2.7.6.1. zsh

2.2.7.6.2. emacs

2.2.7.6.3. xorg-x11-xauth

2.2.8. Created:

2.2.8.1. Ansible administrator account

2.2.8.2. Password-locked, access via "sudo su - ansible-admin"

2.2.8.3. /etc/ansible as home directory

2.2.8.4. RSA keypair

2.2.9. Added ansible-admin public key as a deploy key to ualibraries/ansible-config repository.

2.2.10. Cloned ualibraries/ansible-config to ansible1:/etc/ansible.

2.2.11. Standard status: leave "master" branch checked out.

2.3. Created "openaz-tmpl" virtual machine template in LCU1 vSphere

2.3.1. DNS:

2.3.1.1. openaz-tmpl.library.arizona.edu <=> 10.130.155.13

2.3.1.2. openaz-tst.library.arizona.edu <=> 150.135.174.13

2.3.1.3. openaz-stg.library.arizona.edu <=> 150.135.174.14

2.3.1.4. openaz-prd.library.arizona.edu <=> 150.135.174.15

2.3.1.5. open.uapress.arizona.edu --> openaz-prd.library.arizona.edu

2.3.2. Ubuntu 16.04 (64 bit)

2.3.3. 2 vCPUs, 8 GB RAM, 20 GB storage

2.3.4. LCU1-Test subnet

2.3.5. CIS Ubuntu 16 benchmark filesystem layout

2.3.6. Install as: standard system utilities, OpenSSH server

2.3.7. Added:

2.3.7.1. open-vm-tools

2.3.7.2. python

2.3.8. Created:

2.3.8.1. Ansible administrator account

2.3.8.2. Password-locked

2.3.8.3. Added ansible-admin public key to authorized_keys

2.3.8.4. Added no-password privilege escalation via sudoers

2.3.8.5. Added ping-gateway-on-reboot crontab entry

2.3.9. Converted to virtual machine template in vCenter.

3. Implementation of Ansible configuration

3.1. Vocabulary

3.1.1. Node Types

3.1.1.1. A.k.a. "hosts" in most of the Ansible documentation, which conflicts with ESXi "hosts" in vSphere, etc.

3.1.1.2. "Control Node"

3.1.1.2.1. python

3.1.1.2.2. ansible

3.1.1.2.3. In this case, "ansible1"

3.1.1.3. "Client Node"

3.1.1.3.1. python

3.1.1.3.2. key-based ssh access

3.1.1.3.3. privilege escalation

3.1.1.3.4. in this case:

3.1.1.3.4.1. "openaz-tst"

3.1.1.3.4.2. "openaz-stg"

3.1.1.3.4.3. "openaz-prd"

3.1.2. "Inventory"

3.1.2.1. list of client nodes

3.1.2.2. (optionally) arranged into "groups"

3.1.2.3. one or more

3.1.2.4. usually lives at root of Ansible configuration directory

3.1.3. "Groups"

3.1.3.1. string label

3.1.3.2. used to specify a set of related nodes

3.1.3.3. usually found in inventory files

3.1.4. "Task"

3.1.4.1. single specific action or description of desired state

3.1.4.2. e.g.:

3.1.4.2.1. "user account 'foo' must be present"

3.1.4.2.2. "FQDN for this node must be foo.bar.arizona.edu"

3.1.4.2.3. "this file at that location must contain this text"

3.1.4.3. Ansible comes with a large "library" of "modules" that implement many common tasks

3.1.4.3.1. "fetch" module gets files from remote locations via several protocols

3.1.4.3.2. "apt" module interfaces with apt package management system

3.1.4.3.3. "route53" module manages AWS Route53 DNS entries

3.1.4.4. Writing custom modules is a common extension mechanism for Ansible.

3.1.5. "Handler"

3.1.5.1. special type of task

3.1.5.2. triggered by other tasks that cause changes

3.1.5.3. happen after all other tasks in the play finish

3.1.5.4. e.g., "If you wind up changing the firewall configuration, restart the firewalld service as the last thing you do."

3.1.5.5. specified by "notify" directive in task

3.1.6. "Playbook"

3.1.6.1. A set of configuration specifications, e.g. tasks

3.1.6.2. Differentiated by group, tag, etc.

3.1.6.3. Top-level playbook for an Ansible cluster is usually "site.xml"

3.1.6.4. Best practice breaks up big playbooks into smaller files, which are all "included" into the main site.xml

3.1.7. "Play"

3.1.7.1. maps a set of tasks to a set of nodes

3.1.7.2. usually invoked via "ansible-playbook"

3.1.7.3. "running a play"

3.1.8. "Roles"

3.1.8.1. reusable subsets of plays

3.1.8.2. allow a naming convention

3.1.8.2.1. files

3.1.8.2.2. handlers

3.1.8.2.3. tasks

3.1.8.2.4. templates

3.1.8.3. live under "roles" subdirectory

3.1.8.4. "this node participates in these roles"

3.1.9. "Variables"

3.1.9.1. can be set based on

3.1.9.1.1. node ("host_vars")

3.1.9.1.2. group ("group_vars")

3.1.9.2. can be split into "vars.yml" and "vault.yml"

3.2. Local Details

3.2.1. BSD 2-clause license for simplicity's sake

3.2.2. README file that needs to be expanded/updated/cleaned up

3.2.3. "ansible.cfg" configures defaults for how Ansible does it's work, mostly convention over configuration

3.2.4. "_testing" inventory file

3.2.4.1. will be joined by "_staging", "_production", "_support" in the future

3.2.4.2. underscore is because Mike S. is twitchy about how directory listings sort

3.2.4.3. contains one group ("openaz")

3.2.4.4. with one node ("openaz-tst.library.arizona.edu")

3.2.4.5. matching variables:

3.2.4.5.1. "group_vars/openaz"

3.2.4.5.2. "host_vars/openaz-tst.library.arizona.edu"

3.2.5. "site.yml" -- mostly just a place to include "site_<something>.yml" sub-playbooks

3.2.6. "site_openaz.yml" -- configuration (plays, tasks, roles) for Open Arizona service management

3.2.6.1. When you are called with the "provision" tag:

3.2.6.1.1. Apply the "openaz-p" role

3.2.6.1.2. To nodes in the "openaz" group

3.2.6.2. When you are called with the "configure" tag:

3.2.6.2.1. Apply the "openaz" role

3.2.6.2.2. To nodes in the "openaz" group

3.2.7. "roles/openaz-p"

3.2.8. "roles/openaz"

4. Running It

4.1. Open a local terminal

4.2. Ssh to the control node

4.3. Sudo to the Ansible admin

4.4. Check latest version of configuration

4.5. Build the service

4.5.1. ansible-playbook

4.5.1.1. --ask-vault-pass

4.5.1.2. site.yml

4.5.1.3. --inventory _testing

4.5.1.4. --limit openaz

4.5.1.5. --tags provision

4.5.2. ansible-playbook

4.5.2.1. --ask-vault-pass

4.5.2.2. site.yml

4.5.2.3. --inventory _testing

4.5.2.4. --limit openaz

4.5.2.5. --tags configure

4.6. Run it as many times as you want.

4.6.1. Initial runs are about deployment.

4.6.2. Repeated runs are about state maintenance.

4.6.3. Remember: aim for declaration of state, not imperative running of commands.

4.6.4. Yes, "playbook" and "running plays" is confusing for this (Chef is even worse).

5. Things Left to Do

5.1. Fix silly known_hosts stuff.

5.2. Figure out NFS mounts to preserve state between rebuilds.

5.3. Expand configuration to lock down more things.

5.3.1. Firewall

5.3.2. Hosts Allow/Deny

5.3.3. Accounts

5.3.4. Sudoers

5.3.5. CIS Benchmarks

5.4. Maybe move configuration items common to all services to a "_common" role.

5.5. Implement HTTPS for Manifold.

5.6. Think about automating the automation -- Rundeck or something similar.

5.7. Pipe events to somewhere in Slack.

5.8. Put ansible1 under configuration management.

5.9. Put openaz-tmpl under configuration management.

NAMING CONVENTION:
TYPES OF ENVIRONMENTS

can refer to

virtual machines

service environments

other resources - storage, databases, virtual clusters, AWS service objects, etc.

types of inventory

- **"production" ("PRD", "-prd")**
  - **the "real" service**
  - **publically visible, or a dependency of something publically visible**
  - **endusers care that it keeps working**
  - **if we lost it irretrievably, that would be pretty bad**
- **"staging" ("STG", "-stg")**
  - **the new version of the service about to be deployed into production**
  - **the place where QA/UAT takes place**
  - **not usually publically visible**
  - **sometimes used for destructive functional testing prior to production**
  - **service interruption blocks deployment, but doesn't inconvenience end users**
- **"testing" ("TST", "-tst")**
  - **the place where integration testing of new versions/functionality takes place**
  - **might be broken at any given moment, but hopefully not**
  - **almost never publically visible**
- **"support"**
  - **kind of like production for back-office technologists**
  - **we care if it falls over, but it's not particularly visible to service owners or users**
  - **short service interruptions are probably not a big deal**

complete loss in a disaster would be a bad thing

provides sideband services

log aggregation/analysis

availability monitoring

workflow automation

developer or platform tooling

etc.

"scratch"

everything temporary

that we don't really care about

where we're just experimenting or playing around