Developing Ansible Roles for OpenAFS

Michael Meffie, Sine Nomine Associates

August 19, 2019

Intro

- Principal Software Engineer at Sine Nomine Associates
- We do enterprise grade commercial support for OpenAFS
- This talk is about our experiences in going from 0 to 60 with Ansible

Roles:

https://github.com/openafs-contrib/ansible-openafs

OpenAFS

Elevator pitch

- what if your users never needed to install software?
- what if your developers could deploy new versions atomically?
- what if no matter which machine you are on, your files, in the same place.
- single sign-on read and write access
- move data between servers as needed during the day

OpenAFS

- Distributed, caching, network filesystem ** aggressive file caching (callbacks) ** uniform paths ** read-only replication
- Clients for solaris, linux, BSDs, macos, windows
- Servers normally run on linux, BSD, linux
- CMU Research project in late 80's
- IBM Commercial software in 90's
- Open source since 2000
- Traditionally, setup of a new site can take a new person days or weeks!

goals

- easy, fast deployment of OpenAFS servers and clients
- avoid homebrew tools
- initial usage: lab
    - do not preclude production
- multi platform support
    - start with rhel7
    - added debian/ubuntu
    - add more later
- secure
    - no default passwords
    - selinux enforcing by default
    - proper firewall rules

components

- Kerberos server (kdc) and Kerberos clients
- OpenAFS dbservers; one or three
- OpenAFS fileservers; one up to 255
- OpenAFS clients; one or more ** kernel module: afs filesystem driver

ansible leaning curve

- Start with ad-hoc commands
- Single use playbooks
- Roles and playbooks
- Reusable roles

install modes

- Use packages if available (debian/ubuntu)
- Sine Nomine YUM repo
- Custom YUM repo (testing)
- Build and install from source code

roles

```
openafs_krbserver -- Kerberos primary KDC
openafs_krbclient -- kinit
openafs_server    -- db and fileservers
openafs_client    -- afs kernel module
openafs_cell      -- root volumes
openafs_devel     -- install devel tools, build
openafs_robotest  -- install test suite
```

constraints

Some things to move this along for now.

- limit ansible features
- defer tags
- defer role dependencies
- target ansible versions installed by os
- no yaml shortcuts

challenges

- OpenAFS and Kerberos are not so common, so no ansible modules ** resort to shell commands for now
- Supporting old OpenAFS versions requires different setup process
- We dont know ansible ** some trial and error while learning
- The cell "role"
- The build from source process
- How to publish multi-role repo with ansible-galaxy?

tips

- Have an easy way to create and nuke guests ** with proper dns lookup by hostname ** avoid dns loops ** reuse mac addresses to avoid filling dhcp leases
- Study roles in github ** Hadoop roles were helpful for me
- Use variables for indirection
- Use -vv, -vvv, -vvvv and debug tasks
- Always provide defaults
- Put platform specific stuff in `vars/<os-name>.yaml`
- Always prefix variables with role name

ansible pet peeves

- ansible debugging is sub-optimal, esp on retry loops
- ansible logging?
- lots and lots of tiny files, all look alike.
- no variable namespacing
- host variables vs non-host variables
- ansible "role" name ** weird sports metaphor? ** yaml fragment included *once* ** not a library, not a function (except when it is) ** kind of a template?

Future

- Fix non-idempotent tasks; modules?
- Finish test suite setup and running
- Add more platforms
- Publish on Galaxy

Thanks

Questions?