Los Alamos
NATIONAL LABORATORY

EST. 1943

Los Alamos NATIONAL LABORATORY

EST. 1943

Delivering science and technology
to protect our nation
and promote world stability

Configuration Management

…with Ansible

CSCNSI

Configuration Management

Configuration Management

"The discipline of applying technical and administrative direction and surveillance to identify and document the functional and physical characteristics of a *configuration item,* control changes to those characteristics, record and report change processing and implementation status, and verify compliance with specified requirements." --IEEE-Std-610

Configuration Management

- *Ad-Hoc – control what is convenient*: "by hand" modification of a host's software stack to achieve a desired result

```
# vi /etc/dhcp/dhcpd.conf
```

Configuration Management

- *Ad-Hoc – control what is convenient*: "by hand" modification of a host's software stack to achieve a desired result

  ```
  #  vi /etc/dhcp/dhcpd.conf
  ```

- *Partial – control a few things*: use some gizmo as a means to the same end, where ...
  - Source of the configuration change is external to the host software stack
  - Desired result can be verified and re-achieved

Configuration Management

- *Complete – full prescription*: use a gizmo (or gizmos) to somehow specify everything about the configuration of a host

Configuration Management

- *Complete – full prescription*: use a gizmo (or gizmos) to somehow specify everything about the configuration of a host
  - Deterministic: know what your host is running

Configuration Management

- *Complete – full prescription*: use a gizmo (or gizmos) to somehow specify everything about the configuration of a host
  - Deterministic: know what your host is running
  - Reproducible: bare-metal {re}install

Configuration Management

- *Complete – full prescription*: use a gizmo (or gizmos) to somehow specify everything about the configuration of a host
  - Deterministic: know what your host is running
  - Reproducible: bare-metal {re}install
  - Convergent: recovery from unforeseen events

Configuration Management

- *Complete – full prescription*: use a gizmo (or gizmos) to somehow specify everything about the configuration of a host
  - Deterministic: know what your host is running
  - Reproducible: bare-metal {re}install
  - Convergent: recovery from unforeseen events
    - Implies the gizmo can validate the configuration state and take corrective action
    - Implies the gizmo can be run on a regular basis

Why we do CM

Why we do CM

Because we hate system administration!

Why we do CM

Because we hate system administration!

Also, scale.

Why we do CM

- The CMbot never tires
- The CMbot never forgets what it has been taught
- The CMbot's intelligence is the sum of *everyone’s* expertise
- The CMbot is scalable to many nodes and architectures
- The CMbot increases human performance/reliability, freeing us up to do the things we are paid to do *and* enjoy

Intro to Ansible

Ansible is…

- … a configuration management tool
- … an automation tool
- … written in python
- … open-source
- … led out of Red Hat

Ansible's Philosophy

- Ansible performs tasks
  - install a package, copy a file, enable a service, etc.
- Tasks are run on hosts
  - ba-master.lanl.gov
- Related tasks are grouped into roles
  - "slurm server": install slurm package, copy slurm config files, enable slurm service
- Plays assign roles to hosts and run the corresponding tasks
  - ba-master.lanl.gov is a slurm server
- Playbooks orchestrate groups of plays
  - Update master node, then update compute node images, then copy images to service nodes

Tasks

- A base unit of work that needs to be done
  - copy a file
  - install an RPM
  - enable a service
  - create a cron job
  - … and many more
- Tasks are performed by Ansible **modules**
  - copy
  - yum
  - systemd
  - cron
  - … and many more

Example Task: Install and Enable ntp

- Goal:
  - Install the ntp package
  - Enable and start the ntpd service
- On a RHEL7 system:
  - `yum install ntp`
  - `systemctl enable ntpd`
  - `systemctl start ntpd`

Example Task: Install and Enable ntp

- Two Ansible tasks: install the package, enable the service
- Ansible invokes a module to do each task
  - No Linux commands in the tasks
- Tasks generally define the desired state, the module takes care of enforcing it
- Tasks should be idempotent: running multiple times will not change the result

roles/ntp/tasks/main.yaml

```
# Use the ‘package’ module to
# ensure the ntp package is
# installed
- name: "install ntp package"
  package:
    name: ntp
    state: present

# Use the ‘service’ module to
# ensure the service is enabled and
# started
- name: "enable ntpd service"
  service:
    name: ntpd
    state: started
    enabled: yes
```

Example Task: Drop /etc/ntp.conf in place

- Goal:
  - Copy our customize ntp.conf file to /etc/ntp.conf
  - Make the file owned by root:root
  - Set the file's permissions to 0444

- On a RHEL7 system

```
$ cp ntp.conf /etc/ntp.conf
$ chown root:root /etc/ntp.conf
$ chmod 0444 /etc/ntp.conf
```

ntp.conf

```
# Cluster NTP servers
server 204.121.3.1 prefer
server 204.121.6.1
server 127.127.1.0
fudge  127.127.1.0 stratum 10
```

Task: Drop a config file in place

- Uses the copy module
- Copies a file from the repository to a location on the filesystem
- Sets specified permissions and ownership in the process
- Default source: roles/rolename/files/src

roles/ntp/tasks/main.yaml

```yaml
- name: "install ntp config file"
  copy:
    src: ntp.conf
    dest: /etc/ntp.conf
    owner: root
    group: root
    mode: 0444
```

Templates

- Static files aren't always sufficient
- Templates embed variables inside files
- Ansible uses the Jinja2 templating engine to process these files
- Templates can do things like…
  - simple variable substitution
  - loops over lists of variables
  - include other files
  - much more complex actions

Task: Drop a templated config file in place

- Example one:
  - "ntp_server" is a variable
    - ntp_server = 204.121.3.1
  - Jinja2 replaces the variable between {{ and }} with its value
  - Everything else in the file is left alone

Static:
roles/ntp/files/ntp.conf

```
# Cluster NTP servers
server 204.121.3.1
server 127.127.1.0
fudge  127.127.1.0 stratum 10
```

Templated:
roles/ntp/templates/ntp.conf.j2

```
# Cluster NTP servers
server {{ ntp_server }}
server 127.127.1.0
fudge  127.127.1.0 stratum 10
```

Task: Drop a templated config file in place

- Example two:
  - "ntp_servers" is a list of ntp servers
  - Jinja2 interprets the expression between {% and %} as a control structure
  - Control structure syntax is very similar to python syntax

Static:
roles/ntp/files/ntp.conf

```
# Cluster NTP servers
server 204.121.3.1
server 204.121.6.1
server 127.127.1.0
fudge  127.127.1.0 stratum 10
```

Templated:
roles/ntp/templates/ntp.conf.j2

```
# Cluster NTP servers
{% for ip in ntp_servers %}
server  {{ ip }}
{% endfor %}
server 127.127.1.0
fudge  127.127.1.0 stratum 10
```

Task: Drop a templated config file in place

- The template module:
  - Reads the template file
  - Runs the content through the template engine
  - Writes the result to the specified destination
- Default source: roles/rolename/templates/

main.yaml

```
- name: "install ntp config file"
  template:
    src: ntp.conf.j2
    dest: /etc/ntp.conf
    owner: root
    group: root
    mode: 0444
```

Variables

- Can be set…
  - At the host level
  - At the group level
  - In roles, tasks, playbooks, and several other places

- Precedence (greatest to least):
  - host-specific
  - other groups
  - "all" group
  - role defaults

inventory/host_vars/kit-master.lanl.gov/main.yaml

```
cluster_master_hostname: 'kit-master'
```

inventory/group_vars/ccstar/main.yaml

```
ntp_servers:
 - '128.165.4.4'
 - '128.165.4.33'
```

Hosts

- **Hosts** are individual systems that Ansible knows about
- Examples
  - ba-master.lanl.gov
  - kit-master.ccstar.lanl.gov
- The Ansible client …
  - can be run locally on a host
  - can be run on a central systems that connects to the host via ssh

Groups

- **Groups** are names that can be used to target a set of hosts
- Examples
  - turquoise, ccstar, yellow
  - masters, logins, computes
  - badger, kit
  - room341, room205, room270
- Hosts can be members of multiple groups
- All hosts are members of an implicit “all” group

The Inventory

- Defines:
  - Hosts
    - In this example: kit-master, ba-master
  - Groups
    - In this example: ‘masters’, ‘turquoise’, and ‘ccstar’
- Hosts can be members of many groups
  - kit-master is a member of ‘masters’, ‘ccstar’, and ‘all’
  - ba-master is a member of ‘masters’, ‘turquoise’, and ‘all’

inventory/hosts

```
[masters]
kit-master.ccstar.lanl.gov
ba-master.lanl.gov

[turquoise]
ba-master.lanl.gov

[ccstar]
kit-master.ccstar.lanl.gov
```

Roles

- **Roles** combine related tasks into reusable building blocks
- Examples:
  - webserver
  - slurm-master
  - slurm-client
  - mysql
  - ssh
- Assigned to hosts or groups that need a role’s functionality
  - ba-master needs “slurm-master” and “ssh”
  - ba-fe1 needs “slurm-client” and “ssh”

Example Role: NTP

- tasks/
  - Contains yaml files that define the tasks for this role
- templates/
  - Contains templates for this role
- handlers/
  - Contains callbacks that affect components managed by this role
- defaults/
  - Defines default values for variables used in this role

roles/ntp/

```
  tasks/
    main.yaml
  templates/
    ntp.conf.j2
  handlers/
    main.yaml
  defaults/
    main.yaml
```

Plays

- Assign individual tasks to hosts (or groups)
- Assign roles to hosts (or groups)

master-roles.yaml

```
- hosts: master
  roles:
  - common
  - nfs
  - ntp
  - slurm
```

master-tasks.yaml

```
- hosts: master
  tasks:
  - name: "install ntp package"
    package:
      name: ntp
      state: present
  - name: "enable ntpd service"
    service:
      name: ntpd
      state: started
      enabled: yes
```

Playbooks

- Sequences of plays to be run in order
- Can support orchestration workflows
  - Build a master
  - Next, build frontend images
  - Finally, build compute images
- Can support more complex orchestration workflows
  - Build a web server, then build a database server, then configure the webserver to use the database server
- Our environment probably isn’t complex enough to need very complex playbooks
  - Our dependencies are linear
  - Very few “work on host A, then host B, and then host A again” workflows

Anatomy of a Repository

```
ansible.cfg                        # Config file for ansible commands
cluster-masters.yaml               # Playbook file that covers all master nodes
inventory/                         # Inventory directory.  Contains system information.
    group_vars/                    # Group-specific variable definitions
        all                        # Variables applied to all systems
        ccstar                     # Variables applied to systems in the ‘ccstar’ group
        turquoise                  # Same, for the ‘turquoise’ group
    hosts.ccstar                   # Inventory of hosts on the ccstar network
    hosts.turquoise                # Inventory of hosts on the turquoise network
    host_vars/                     # Host-specific variable definitions
        ba-master.lanl.gov/        # Variables that apply only to ba-master
        kit-master.ccstar.lanl.gov/    # Variables that apply only to kit-master
roles/                             # Definitions of role building blocks
    ntp/                           # The ‘ntp’ role.  Provides everything needed for ntp
        defaults/                  # Files that define default values of template variables
            main.yaml              #
        handlers/                  # Files that define handlers for callbacks used in tasks
            main.yaml              #   Handlers do things like restart services when needed
        tasks/                     # Files that define tasks
            main.yaml              #   Tasks do things like copy files and install packages
        templates/                 # Files that define templates
            ntp.conf.j2            #   Templates look like config files, but with variables
    slurm/                         # The ‘slurm’ role.  Currently empty, but similar in
                                   #   concept to the ‘ntp’ role
```

How Ansible Runs

- Run targeting kit-master
  - `ansible-playbook -l kit-master.ccstar.lanl.gov cluster-masters.yaml`
- Inventory file gets read in
  - `kit-master` is a member of masters, ccstar, and all groups
- `cluster-masters.yaml` is read in
  - the masters group is assigned the ntp role
- Tasks in the ntp role are read in
- Each task is run
  - Tasks are performed by the module specified in the task
- Handlers are run
- Done!

Running Ansible - The Commands

- Commands
  - `ansible-playbook` – Run one or more tasks via a playbook
  - `ansible` – Run a single task via the command line
  - `ansible-inventory` – Displays the compiled inventory
  - `ansible-vault` – Encrypts/decrypts files in the Ansible repository
  - `ansible-pull` – Perform a git (or svn) pull (or update) before running locally
  - `ansible-galaxy` – Interacts with Ansible Galaxy, an online role repository
- Plus some extras: ansible-config, ansible-console, ansible-doc

Running Ansible - *Safely*

- Ansible has safety options
  - `--check`
    - Do a dry run. Don't make any changes, but reports what tasks would have made changes
  - `--diff`
    - When changing a file, display a diff between the existing file and the new file
  - `--step`
    - Run interactively, asking for confirmation before running each task
  - `--syntax-check`
    - Do a sanity check of a playbook without running it

Running Ansible - Example Command Lines

- Run a playbook
  - `ansible-playbook -i inventory/ -l localhost all.yaml`
- Run a playbook in check mode
  - `ansible-playbook --check -i inventory/ -l localhost all.yaml`
- Run a playbook in interactive mode
  - `ansible-playbook --step -i inventory/ -l localhost all.yaml`
- Display a list of all variables Ansible knows about for a host
  - `ansible-inventory -i inventory/ --host kit-master.ccstar.lanl.gov`

Exercises: Homework

- Write tasks to manage some common packages
  - Install `bind-utils`, `git`, and `zsh`
  - It probably makes sense to do this in the `common` role
- Write a task to manage `/etc/motd`
  - Create a role that manages a static `/etc/motd` file
- Write a role to manage rsyslog
  - Install `rsyslog` package
  - Enable `rsyslog` service
  - Install `/etc/rsyslog.conf`

Questions?