Linux Academy

Ansible Quick Start

Just the facts...

Stosh Oldham

stosh@linuxacademy.com

May 29, 2019

Contents

Install and Configure Ansible

- In order to install Ansible, you must configure the EPEL repository on your system.
- Once the EPEL repository is configured, your package manager installs Ansible and manage dependencies.
    - `sudo yum install ansible`
- It is advisable to install some means of source control as well.
    - `sudo yum install git`
- Configuration Files:
    - `/etc/ansible/ansible.cfg`:
        - The primary Ansible configuration file.
        - Notable configurations include:
            - Default inventory configuration.
            - Default remote user.
    - `/etc/ansible/hosts`:
        - Default Ansible Inventory File.
        - An inventory is a list of hosts that Ansible manages.
        - Inventory location may be specified as follows:
            - Default: `/etc/ansible/hosts`.
            - Specified by CLI: `ansible -i <filename>`.
            - Can be set in `ansible.cfg`.
            - Example Inventory file:

```
mail.example.com ansible_port=5556 ansible_host=192.168.0.10

[webservers]
httpd1.example.com
httpd2.example.com

[labservers]
lab[01:99]
```

            - The first line defines a host `mail.example.com`.
            - Two variables are affiliated with the host, `ansible_port` and `ansible_host`.
            - The group's web servers and lab servers are defined in this example.
            - Note the lab servers group has 99 hosts in it that are defined via a pattern.
            - The expression `lab[01:03]` is the same as specifying `lab01`, `lab02`, `lab03`.
- Configure SSH users (with `sudo`) and keys:
    - While it is possible to connect to a remote host with Ansible using password authentication using `-k` (note lowercase), it is not a common practice as it can incur significant overhead concerning manual intervention.

- Ansible is best implemented using a common user across all Ansible controlled systems.
- The `ssh-keygen` and `ssh-copy-id` command can facilitate creating a pre-shared key for user authentication.
- `/etc/sudoers` may be edited to allow your selected user to `sudo` any command without a password for the most automated configuration using the line `ansible ALL=(ALL) NOPASSWD: ALL`.
- It is also possible to prompt for a `sudo` password at runtime using `-K`(note uppercase) if desired; this can become a challenge when executing against many systems.

When you need help

- Useful documentation is provided at `docs.ansible.com`.
- A module index is provided at `docs.ansible.com` that provides detailed information on each module.
- Ansible ships with the `ansible-doc` command which:
  - Specifies a module name as a parameter will provide module specific documentation.
  - Uses the `-l` flag to list installed modules with a brief description.

Ansible ad-hoc

- Ansible `ad-hoc` commands are analogous to `bash` commands.
- Playbooks are analogous to bash scripts.
- Syntax: `ansible \<HOST> -b -m \<MODULE> -a "\<ARG1 ARG2 ARGN>" -f \<NUM_FORKS>`:
  - HOST is a host or host group defined in the Ansible inventory file.
  - `b` is for `become`:
    - Replaces the depreciated `-s` flag as in `sudo`.
    - Ansible escalates permission to `--become-user` using the method defined by `--become-method`.
    - Default `become-user` is `root`.
    - Default `become-method` is `sudo`.
  - `m` is for modules for the command to use.
  - `a` is used for parameters to pass, if used without `m`, it is like running a shell command on the target system(s).
  - `f` is used to set forks for parallelism, which is how you can have Ansible execute plays simultaneously on many hosts.
- Common modules with required parameters:
  - `ping`
  - `setup`
  - `yum "name=\<NAME> state=\<STATE>"`
  - `service "name=\<NAME> state=\<STATE>"`
  - `copy "src=\<SOURCE_PATH> dest=\<ABSOLUTE_DESTINATION_PATH>"`

Ansible Playbooks

- Basic Ansible Playbook structure:
  - As `ad-hoc` commands are to `bash` commands, playbooks are to bash scripts.
  - Playbooks are ran using the `ansible-playbook` command, not the `ansible` command.
  - You must specify the playbook to run as a parameter.
  - Playbooks are written in YAML:
    - Each play is an element in a sequence.
    - Plays contain lists of hosts and, at minimum, one or more tasks.
    - Each task has a name and module.
    - Modules may have parameters.
    - Improper indentation can cause a playbook to err in a vague way.
    - Spaces matter.
- Sample playbook:

```
- hosts: webservers
  become: yes

  tasks:
  - name: ensure apache is at the latest version
    yum:
      name: httpd
      state: latest
  - name: write the apache config file
    template:
      src: /srv/httpd.j2
      dest: /etc/httpd.conf

- hosts: databases
  remote_user: root

  tasks:
  - name: ensure postgresql is at the latest version
    yum:
      name: postgresql
      state: latest
  - name: ensure that postgresql is started
    service:
      name: postgresql
      state: started
```

- The above playbook is composed of two plays:
  - The first play installs a package (using the `yum` module) and creates a configuration file from a template (using the `template` module):

        - Note that `become` is set for the first play:
            - This is the same as using `-b` when running an `ad-hoc` command.
            - Ansible will use `sudo` to escalate to the `root` account on the remote system.
            - This is required for the tasks defined.
        - The second play ensures `postgresql` is installed and running using both the `yum` and `service` modules.
        - Note that `become` is ***not*** set in the second play yet the `remote_user` is set to `root`:
            - This is not a particularly good security practice but does demonstrate some basic Ansible configuration.

- Check Mode provides a quick sanity check:
    - Dry run that does not affect the systems.
    - Syntax: `ansible-playbook foo.yml --check`
- Retry file:
    - If a playbook fails, a retry file is generated containing the list of hosts where the play failed
    - A file called `\<playbook-name>.retry` is created upon a playbook failure.
    - The file may be specified using `--limit` with the same playbook to reattempt the playbook at a later time.
- Plays should be safe to run repeatedly against the same target without ill effect.

Ansible Variables

- Variable names should be letters, numbers, and underscores.
- Variables should always start with a letter.
- Examples of valid variable names:
    - `foobar`
    - `foo_bar`
    - `foo5`
- Examples of invalid variable names:
    - `foo-bar`
    - `1foobar`
    - `foo.bar`
- Variables can be scoped by group, host, or within a playbook.
- Variables are passed in via the command line using the `--extra-vars` or `-e` flag or are defined within a playbook:

    - CLI Example: `ansible-playbook service.yml -e "target_hosts=localhost target_service=httpd`
    - Playbook Example:

```
hosts: webservers
become: yes
vars:
  target_service: httpd
  target_state: started
tasks:
  - name: Ensure target service is at target state
    service:
      name: "{{ target_service }}"
      state: "{{ target_state }}"
```

- Variables are referenced using double curly braces.
- It is good practice to wrap variable names or statements containing variable names in weak quotes:
    - Example: `hosts: "{{ my_host_var }}"`
- Typical uses of variables:
- Customize configuration values.
- Hold return values of commands.
- Ansible has many default variables for controlling Ansible's behavior.

Ansible Facts

- Ansible facts are simply various properties regarding a given remote system.
- The `setup` module can retrieve facts:
    - The filter parameter takes regex to allow you to prune fact output.
- Facts are gathered by default in Ansible playbook execution:
    - The keyword `gather_facts` may be set in a playbook to change fact gathering behavior.
- It is possible to print Ansible facts in files using variables.
- Facts may be filtered using the setup module ad-hoc by passing a value for the `filter` parameter.
- Ansible command output may be directed to a file using the `--tree outputfile` flag which may be helpful when working with facts.
- It is possible to use `{{ ansible_facts }}` for conditional plays based on facts.

Debugging in Ansible

- The `debug` module may be used to help troubleshoot plays:
  - Used to print detail information about in-progress plays.
  - Handy for troubleshooting.
- Debug takes two primary parameters that at mutually exclusive:
  - `msg`: A message that is printed to STDOUT
  - `var`: A variable whose content is printed to STDOUT.
  - Example:

```
- debug:
    msg: "System {{ inventory_hostname }} has uuid {{ ansible_product_uuid }}"
```

- The `register` module is used to store task output:
  - It essentially can save results of a command.
  - Several attributes are available: return code, stderr, and stdout.
  - Example:

```
- hosts: all
  tasks:
  - shell: cat /etc/motd
    register: motd_contents
  - shell: echo "motd contains the word hi"
    when: motd_contents.stdout.find('hi') != -1
```

Notifications and Handlers

- Ansible provides a mechanism that allows an action to be flagged for execution when a task performs a change.
- By only executing certain tasks during a change, plays are more efficient.
- This mechanism is known as a ***handler*** in Ansible.
- A handler may be called using the `notify` keyword.
- No matter how many times a handler is flagged in a play, it is only ran once during a play's final phase.
- `notify` will only flag a handler if a task block makes changes:
  - Example:

```
- name: template configuration file
  template:
    src: template.j2
    dest: /etc/foo.conf
  notify:
     - restart memcached
     - restart apache
```

- The calls made in the notify section correspond to handler definitions within the play.
- A handler may be defined similarly to tasks:
    - Example:

```
handlers:
- name: restart cache service
   service:
     name: memcached
     state: restarted
   listen: "restart memcached"
- name: restart web services
   service:
    name: apache
    state: restarted
  listen: "restart apache"
```