JINJA2 TEMPLATES

Describing Jinja2 Templates

Objectives

After completing this section, students should be able to:

- Describe Jinja2 templates
- Describe the differences between YAML files and Jinja2 templates
- Describe variables, control structures, and comment use in Jinja2 templates

Introduction to Jinja2

Ansible uses the Jinja2 templating system to modify files before they are distributed to managed hosts. Generally speaking, it is preferable to avoid modifying configuration files through logic in templates. However, templates can be useful when systems need to have slightly modified versions of the same file. Ansible also uses Jinja2 to reference variables in playbooks.

Ansible allows Jinja2 loops and conditionals to be used in templates, but they are not allowed in playbooks. Ansible playbooks are completely machine parsable YAML. This is an important feature, because it means it is possible to have code generate pieces of files, or to have other third-party tools read Ansible files. Not everyone will need this feature, but it can unlock interesting possibilities.

Delimiters

Variables or logic expressions are placed between tags, or delimiters. For example, Jinja2 templates use {% EXPR %} for expressions or logic (for example, loops), while {{ EXPR }} are used for outputting the results of an expression or a variable to the end user. The latter tag, when rendered, is replaced with a value or values, and are seen by the end user. Use {# COMMENT #} syntax to enclose comments.

In the following example the first line includes a comment that will not be included in the final file. The variable references in the second line are replaced with the values of the system facts being referenced.

```
{# /etc/hosts line #}
{{ ansible_default_ipv4.address }}    {{ ansible_hostname }}
```

Control Structures

Often the result of a play depends on the value of a variable, fact (something learned about the remote system), or previous task result. In some cases, the values of variables depend on other variables. Further, additional groups can be created to managed hosts based on whether the hosts match other criteria. There are many options to control execution flow in Ansible.

Loops

Jinja2 uses the for statement to provide looping functionality. In the following example, the user variable is replaced with all the values included in users.

Conditionals

Jinja2 uses the if statement to provide conditional control. In the following example, the result is displayed if the finished variable is True.

```
{% if finished %}
    {{ result }}
{% endif %}
```

Variable Filters

Jinja2 provides filters which change the output format for template expressions (for example, to JSON). There are filters available for languages such as YAML or JSON. The **to_json** filter formats the expression output using JSON, and the **to_yaml** filter uses YAML as the formatting syntax.

```
{{ output | to_json }}
{{ output | to_yaml }}
```

The expressions used with when clauses in Ansible playbooks are Jinja2 expressions. Built-in Ansible filters that are used to test return values include **failed, changed, succeeded**, and skipped. The following task shows how filters can be used inside of conditional expressions.

```
tasks:
... Output omitted ...
  - debug: msg="the execution was aborted"
    when: returnvalue | failed
```

Issues to be Aware of with YAML vs. Jinja2 in Ansible

The use of some Jinja2 expressions inside of a YAML playbook may change the meaning for those expressions, so they require some adjustments in the syntax used.

1. YAML syntax requires quotes when a value starts with a variable reference ({{ }}). The quotes prevent the parser from treating the expression as the start of a YAML dictionary. For example, the following playbook snippet will fail:

```
- hosts: app_servers
  vars:
    app_path: {{ base_path }}/bin
```

Instead, use the following syntax:

```
- hosts: app_servers
  vars:
    app_path: "{{ base_path }}/bin"
```

References

Template Designer Documentation — Jinja2 Documentation
http://jinja.pocoo.org/docs/dev/templates/
template - Templates a file out to a remote server — Ansible Documentation
http://docs.ansible.com/ansible/template_module.html
Variables — Ansible Documentation
http://docs.ansible.com/ansible/playbooks_variables.html

Guided Exercise: Implementing Jinja2 Templates

#cd /home/ansible/playbook/jinja2

Create the **inventory** file in the current directory. This file configures two groups: *webservers* and *workstations*. Include the system ***robo*** in the *webservers* group, and the system *robo.2.0* in the *workstations* group.

```
[ansible@robo jinja2]$ cat inventory
[webservers]
robo

[workstations]
robo2.0
[ansible@robo jinja2]$
```

Create a template for the Message of the Day. Include it in the **motd.j2** file in the current directory. Include the following variables in the template:
• **ansible_hostname** to retrieve the managed host hostname.
• **ansible_date_time.date** for the managed host date.
• **system_owner** for the email of the owner of the system. This variable needs to be defined with an appropriate value in the **vars** section of the playbook template.

```
[ansible@robo jinja2]$ cat motd.j2
This is the system {{ ansible_hostname }}.
Today's date is: {{ ansible_date_time.date }}.
Only use this system with permission.
You can ask {{ system_owner }} for access.
[ansible@robo jinja2]$
```

Create a playbook in a new file in the current directory, named **motd.yml**. Define the **system_owner** variable in the **vars** section, and include a task for the *template* module, which maps the **motd.j2** Jinja2 template to the remote file **/etc/motd** on the managed hosts. Set the owner and group to *root*, and the mode to *0644*.

cat motd.yml

```
[ansible@robo jinja2]$ cat motd.yml
---
- hosts: all
  user: ansible
  become: yes
  vars:
    system_owner: deepan.redhat@gmail.com
  tasks:
    - template:
        src:  motd.j2
        dest: /etc/motd
        owner: root
        group: root
        mode: 0644
```

ansible-playbook --syntax-check motd.yml

ansible-playbook motd.yml

```
[ansible@robo jinja2]$ ansible-playbook motd.yml
SUDO password:

PLAY [all] *********************************************************************

TASK [Gathering Facts] *********************************************************
ok: [robo2.0]
ok: [robo]

TASK [template] ****************************************************************
changed: [robo]
changed: [robo2.0]

PLAY RECAP *********************************************************************
robo                       : ok=2    changed=1    unreachable=0    failed=0
robo2.0                    : ok=2    changed=1    unreachable=0    failed=0
```

Log in to robo using the ansible user, to verify the motd is displayed when logging in. Log out when you have finished.

```
[ansible@robo jinja2]$ ssh robo
Last login: Thu Apr 18 18:25:42 2019 from robo
This is the system robo.
Today's date is: 2019-04-18.
Only use this system with permission.
You can ask deepan.redhat@gmail.com for access.
```

```
[ansible@robo jinja2]$ ssh robo2.0
Last login: Thu Apr 18 18:25:42 2019 from robo
This is the system robo2.
Today's date is: 2019-04-18.
Only use this system with permission.
You can ask deepan.redhat@gmail.com for access.
```