Ansible Jinja2 Template

By: Er. Vikas Nehra (M. Tech, B. Tech), Experience: 15 + Years

Session - 21 Agenda:

1. Templating (Jinja2)

Templating (Jinja2):

Jinja2 is a powerful and easy to use python-based templating engine that comes in handy in an IT environment with multiple servers where configurations vary every other time. Creating static configuration files for each of these nodes is tedious and may not be a viable option since it will consume more time and energy. And this is where templating comes in.

Jinja2 templates are simple template files that store variables that can change from time to time. When Playbooks are executed, these variables get replaced by actual values defined in Ansible Playbooks. This way, templating offers an efficient and flexible solution to create or alter configuration file with ease.

All templating happens on the Ansible controller before the task is sent and executed on the target machine. This approach minimizes the package requirements on the target (jinja2 is only required on the controller). It also limits the amount of data Ansible passes to the target machine. Ansible parses templates on the controller and passes only the information needed for each task to the target machine, instead of passing all the data on the controller and parsing it on the target.

Template architecture:

A Jinja2 template file is a text file that contains variables that get evaluated and replaced by actual values upon runtime or code execution. In a Jinja2 template file, you will find the following tags:

{{ }} : These double curly braces are the widely used tags in a template file and they are used for embedding variables and ultimately printing their value during code execution. For example, a simple syntax using the double curly braces is as shown: The {{ webserver }} is running on {{ nginx-version }}

{% %} : These are mostly used for control statements such as loops and if-else statements.

{# #} : These denote comments that describe a task.

In most cases, Jinja2 template files are used for creating files or replacing configuration files on servers. Apart from that, you can perform conditional statements such as loops and if-else statements and transform the data using filters and so much more.

Template files bear the .j2 extension, implying that Jinja2 templating is in use.

Let's understand it with an example, suppose we want to display some node specific details like its OS name, IP Address, hostname etc. in form of **Message of The Day (MOTD)** on each node whenever we login to that node. We know that /etc/motd (MOTD) file is used in Linux to display such information at the login prompt through SSH. For this purpose, we write the message in this file which we want to display at the login prompt in form of MOTD.

 # vim /etc/motd

Welcome To Nehra Classes YouTube Channel.

But in case of having a large number of nodes (client machines) in the environment, it would

Ansible Jinja2 Template

By: Er. Vikas Nehra (M. Tech, B. Tech), Experience: 15 + Years

become very hectic to manually editing the MOTD file on each of these nodes. If it would have been the same content, we could have used the copy module or some other module to create/update the file. But here we need machine specific details which we want to display with the help of the MOTD file and there may several such types of tasks where we need machine specific information on the managed nodes, there Ansible Jinja Templates (Jinja2) comes into picture. We can easily perform these tasks with the help of templating (Jinja2).

For using jinja template in ansible we create the jinja file having extension as j2. We can call the variables, conditionals and filters with some special characters in this file.

$ vim motd.j2

```
********Welcome to {{ ansible_distribution }} Server.********
The IP address of this machine is {{ ansible_default_ipv4.address }}
The machine hostname is {{ ansible_fqdn }}
******NEHRA CLASSES ARE AWESOME.******
```

Now, we have to create an ansible playbook to push this template to the managed nodes.

$ vim jinja.yml

```
---
- name: MOTD file creation playbook.
  hosts: RHEL
  become: true
  tasks:
    - name: Copying the MOTD file from Ansible server to the managed nodes.
      template:
            src: /home/vikasnehra/motd.j2
            dest: /etc/motd
...
```

Let's check the /etc/motd file first on the managed nodes.

$ ansible all -m command -a 'cat /etc/motd'

Now, we can execute the playbook to push the task, which will copy the machine specific desired information in the /etc/motd file in the managed nodes.

$ ansible-playbook jinja.yml

Verify the task using ansible ad-hoc command.

$ ansible all -m command -a 'cat /etc/motd'

Let's take one more example to configure the Apache web server on the managed nodes. To get a better sense of how you can push configuration files, we are going to create a Jinja2 template that creates an index.html file in the web root or document directory /var/www/html.

$ vim index.j2

```
<html>
   <center><h1> The Apache webserver is running on {{ ansible_hostname }} </h1>
   </center>
</html>
```

Ansible Jinja2 Template

By: Er. Vikas Nehra (M. Tech, B. Tech), Experience: 15 + Years

Let's create the ansible playbook to configure the Apache web server using this template.

$ vim apache.yml

```
---
- name: Apache Web Server Configuration Playbook
  hosts: RHEL
  become: true
  tasks:
    - name: Installing Apache Latest Packages
      yum:
        name: httpd
        state: latest
    - name: Starting & Enabling Apache HTTPD Service
      service:
        name: httpd
        state: started
        enabled: true
    - name: Setting Up Apache Web Server
      template:
        src: /home/vikasnehra/index.j2
        dest: /var/www/html/index.html
    - name: Restarting Apache HTTPD Service
      service:
        name: httpd
        state: restarted
    - name: Allowing HTTP Traffic in the Firewall.
      firewalld:
        service: http
        permanent: yes
        state: enabled
    - name: Reload the Firewall Service
      service:
        name: firewalld
        state: reloaded
...
```

Execute the playbook.

$ ansible-playbook apache.yml

It may throw an error because of missing firewalld module, which we can install from ansible galaxy. Install the additional modules from the ansible galaxy using below command.

$ ansible-galaxy collection install ansible.posix

Now, we can execute the ansible playbook to install the Apache web server in the managed nodes.

$ ansible-playbook apache.yml

Verify the working of the Apache web server.

$ curl node1

Ansible Jinja2 Template

By: Er. Vikas Nehra (M. Tech, B. Tech), Experience: 15 + Years

$ curl node3

Or use any web browser and mention the IP address of the managed node there, it will show you the page having machine specific information.

Jinja2 template with Conditionals:

Jinja2 templating can also be used with conditional statements such as for loops to iterate over a list of items. Consider the Playbook jinja_conditionals.yml, We are going to create a template that will iterate over the list of fruit names called 'fruits' and print the result in the fruits.txt destination file on the managed nodes.

$ vim jinja_conditionals.yml

```
---
- name: Ansible jinja2 template with conditionals example playbook
  hosts: RHEL
  vars:
     fruits:
       - mango
       - apple
       - grapes
       - banana
       - litchi
       - papaya
  tasks:
     - name: Using loop with jinja2 template
       template:
          src: /home/vikasnehra/fruits.j2
          dest: /home/vikasnehra/fruits.txt
...
```

Let's create the jinja2 template file for the above task.

$ vim fruits.j2

```
The list of fruit names is as follows:
{% for item in fruits %}
{{ item }}
{% endfor %}
```

Now, we can execute the playbook.

$ ansible-playbook jinja_conditionals.yml

Verify the task with the ansible ad-hoc command.

$ ansible RHEL -m command -a 'cat /home/vikasnehra/fruits.txt'

If you will put one line space in the jinja2 file same will reflect in the output file after the execution of the playbook.

$ vim fruits.j2

```
The list of fruit names is as follows:
```

{% for item in fruits %}

{{ item }}

{% endfor %}

Now, we can execute the playbook.
$ ansible-playbook jinja_conditionals.yml

Verify the task with the ansible ad-hoc command.
$ ansible RHEL -m command -a 'cat /home/vikasnehra/fruits.txt'

Jinja2 template with filters:
Filters are used to alter the appearance of output or formatting data. This works by piping the variable name as shown:
{{ variable | argument }}

Let's check out a few use cases:
a) Transform strings into either Uppercase or lowercase format:
For example, to print the values in the previous list in uppercase characters using the template, pipe the variable item into the 'UPPER' argument as shown: {{ item | upper }}
$ vim fruits.j2
The list of fruit names is as follows:
{% for item in fruits %}
{{ item | upper }}
{% endfor %}

Now, we can execute the playbook.
$ ansible-playbook jinja_conditionals.yml

Verify the task with the ansible ad-hoc command.
$ ansible RHEL -m command -a 'cat /home/vikasnehra/fruits.txt'

You can see that, when the playbook is executed, the values are transformed into uppercase.

If the values are in lowercase from the start, use the 'lower' argument. {{ item | lower }}
$ vim fruits.j2
The list of fruit names is as follows:
{% for item in fruits %}
{{ item | lower }}
{% endfor %}

Now, we can execute the playbook.
$ ansible-playbook jinja_conditionals.yml

Verify the task with the ansible ad-hoc command.
$ ansible RHEL -m command -a 'cat /home/vikasnehra/fruits.txt'

Ansible Jinja2 Template

By: Er. Vikas Nehra (M. Tech, B. Tech), Experience: 15 + Years

You can see that, when the playbook is executed, the values are transformed into lowercase.

b) Use list filters to display maximum & minimum values:
If you are working with arrays or lists inside the template as shown, you can choose to print out your preferred values based on certain criteria.

$ vim numbers.yml

```
---
- name: Ansible jinja2 template with filters example playbook
  hosts: RHEL
  tasks:
     - name: Using filters with jinja2 template
       template:
           src: /home/vikasnehra/numbers.j2
           dest: /home/vikasnehra/numbers.txt
...
```

For example, to print out the minimum value in a list, pass the whole list to the 'min' filter as shown.
{{ [100, 37, 45, 65, 60, 78] | min }} => 37

$ vim numbers.j2

The lowest number is: {{ [100, 37, 45, 65, 60, 78] | min }}

Now, we can execute the playbook.

$ ansible-playbook numbers.yml

Verify the task with the ansible ad-hoc command.

$ ansible RHEL -m command -a 'cat /home/vikasnehra/numbers.txt'

To get the maximum value, use the 'max' filter.
{{ [100, 37, 45, 65, 60, 78] | max }} => 100

$ vim numbers.j2

The lowest number is: {{ [100, 37, 45, 65, 60, 78] | max }}

Now, we can execute the playbook.

$ ansible-playbook numbers.yml

Verify the task with the ansible ad-hoc command.

$ ansible RHEL -m command -a 'cat /home/vikasnehra/numbers.txt'

You can obtain unique values from a list of duplicate values in an array using the unique filter as shown:
{{ [3, 4, 3, 3, 4, 2, 2] | unique }} => 3, 4, 2

$ vim numbers.j2

The unique numbers are: {{ [3, 4, 3, 3, 4, 2, 2] | unique }}

Ansible Jinja2 Template

By: Er. Vikas Nehra (M. Tech, B. Tech), Experience: 15 + Years

c) Replacing a string value with another:

Let's use the same playbook.

$ vim string.yml

```
---
- name: Ansible jinja2 template with filters example playbook
  hosts: RHEL
  tasks:
      - name: Using filters with jinja2 template
        template:
            src: /home/vikasnehra/string.j2
            dest: /home/vikasnehra/string.txt
...
```

Additionally, you can replace a string with a new one using the replace argument as shown:

{{ "Hello Everyone" | replace ("Everyone", "World") }} => Hello World

$ vim string.j2

{{ "Hello Everyone" | replace ("Everyone", "World") }}

Now, we can execute the playbook.

$ ansible-playbook string.yml

Verify the task with the ansible ad-hoc command.

$ ansible RHEL -m command -a 'cat /home/vikasnehra/string.txt'

In the above example, the string guys will be replaced with world and now the statement will read:

Hello world

These are just a few filters. There are tons of built-in filters that you can use to manipulate the output of the Ansible Playbook execution.

The Jinja2 templating is an ideal solution when handling dynamic variables in configuration files. It's a much efficient option than manually changing values which often takes a lot of time and can be quite tedious. Your feedback on this article is much welcome.

Thank You