Ansible Workshop - Exercises

Basics

Get to know Ansible and learn to write your first Ansible Playbooks.

8 - Templating with Jinja2

Objective

This exercise will cover Jinja2 templating. Ansible uses Jinja2 templating to modify files before they are distributed to managed hosts. Jinja2 is one of the most used templating engines for Python, take a look at the documentation for additional information.

Guide

Step 1 - Using Templates in Playbooks

When a template for a file has been created, it can be deployed to the managed hosts using the `template` module, which supports the transfer of a local file from the control node to the managed hosts.

As an example of using templates you will change the motd file to contain host-specific data.

First create the directory `templates` to hold template resources in `~/ansible-files/`:

```
[student@ansible-1 ansible-files]$ mkdir templates
```

Then in the `~/ansible-files/templates/` directory create the template file `motd-facts.j2`:

```
Welcome to {{ ansible_hostname }}.
{{ ansible_distribution }} {{ ansible_distribution_version}}
deployed on {{ ansible_architecture }} architecture.
```

The template file contains the basic text that will later be copied over. It also contains variables which will be replaced on the target machines individually.

Next we need a playbook to use this template. In the `~/ansible-files/` directory create the Playbook `motd-facts.yml`:

```
---
- name: Fill motd file with host data
  hosts: node1
  become: true
  tasks:
    - name: Deploy message of the day file
      ansible.builtin.template:
        src: motd-facts.j2
        dest: /etc/motd
        owner: root
        group: root
        mode: "0644"
```

As we just learned what *handlers* do, let's add one to this playbook. Add the handlers block with a simple task, which just outputs a message:

```yaml
---
- name: Fill motd file with host data
  hosts: node1
  become: true
  handlers:
    - name: Output info about MOTD
      listen: motd_changed_handler
      ansible.builtin.debug:
        msg: "The Message of the Day was updated! SSH to node1 and check the content."
  tasks:
    - name: Deploy message of the day file
      ansible.builtin.template:
        src: motd-facts.j2
        dest: /etc/motd
        owner: root
        group: root
        mode: "0644"

    - name: Add script to /etc/profile.d for MOTD
      ansible.builtin.copy:
        content: cat /etc/motd
        dest: /etc/profile.d/motd.sh
        owner: root
        group: root
        mode: "0755"
```

Before we do a bigger challenge lab, let's see if you remember how handlers are triggered. Currently, the handler is not triggered, add the missing keyword to the task, which deploys the template.

Solution

Add the *notify* keyword and the name of the handler:

```yaml
---
- name: Fill motd file with host data
  hosts: node1
  become: true
  handlers:
    - name: Output info about MOTD
      listen: motd_changed_handler
      ansible.builtin.debug:
        msg: "The Message of the Day was updated! SSH to node1 and check the content."
  tasks:
    - name: Deploy message of the day file
      ansible.builtin.template:
        src: motd-facts.j2
        dest: /etc/motd
        owner: root
        group: root
        mode: "0644"
      notify: motd_changed_handler

    - name: Add script to /etc/profile.d for MOTD
      ansible.builtin.copy:
        content: cat /etc/motd
        dest: /etc/profile.d/motd.sh
        owner: root
        group: root
        mode: "0755"
```

You have done this a couple of times by now:

- Understand what the Playbook does.
- Execute the Playbook `motd-facts.yml`.
- Observe if the handler was triggered. Re-Run the playbook multiple times.
- Login to node1 via SSH and check the message of the day content.
- Log out of node1.

You should see how Ansible replaces the variables with the facts it discovered from the system. The handler was only triggered when the task reported a *changed* state.

Step 2 - Challenge Lab

Add a line to the template to list the current kernel of the managed node.

- Find a fact that contains the kernel version using the commands you learned in the "Ansible Facts" chapter.

> **Tip**
>
> Filter for *kernel*.

Run the newly created playbook to find the fact name.

- Change the template to use the fact you found.
- Run the motd playbook again.
- Check motd by logging in to node1

Solution

Find the fact:

```
---
- name: Capture Kernel Version
  hosts: node1
  tasks:
    - name: Collect only kernel facts
      ansible.builtin.setup:
        filter:
          - '*kernel'
      register: setup_output

    - name: Output variable content
      ansible.builtin.debug:
        msg: "{{ setup_output }}"
```

With the wildcard in place, the output shows:

```
TASK [debug] ********************************************************************
ok: [node1] => {
    "setup": {
        "ansible_facts": {
            "ansible_kernel": "4.18.0-305.12.1.el8_4.x86_64"
        },
        "changed": false,
        "failed": false
    }
}
```

With this we can conclude the variable we are looking for is labeled `ansible_kernel`. Then we can update the motd-facts.j2 template to include `ansible_kernel` as part of its message.

Modify the template `motd-facts.j2`:

```
Welcome to {{ ansible_hostname }}!
Host runs {{ ansible_distribution }} {{ ansible_distribution_version}}
Deployed on {{ ansible_architecture }} architecture
The kernel version is {{ ansible_kernel }}
```

Run the playbook.

Ansible

```
[student@ansible-1 ~]$ ansible-playbook motd-facts.yml
```

Navigator

```
[student@ansible-1 ~]$ ansible-navigator run motd-facts.yml -m stdout
```

Verify the new message via SSH login to `node1`.

▼ Details

```
[student@ansible-1 ~]$ ssh node1
Welcome to node1.
Host runs RedHat 8.1
Deployed on x86_64 architecture
The kernel version is 4.18.0-305.12.1.el8_4.x86_64.
```