Ansible Workshop - Exercises

Basics

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

5 - Work with conditionals

Objective

In a playbook, you may want to execute different tasks, or have different goals, depending on the value of a fact (data about the remote system), a variable, or the result of a previous task. You may want the value of some variables to depend on the value of other variables. Or you may want to create additional groups of hosts based on whether the hosts match other criteria. You can do all of these things with conditionals.

Guide

Ansible can use conditionals to execute tasks or plays when certain conditions are met.

To implement a conditional, the `when` statement must be used, followed by the condition to test. The condition is expressed using one of the available operators like e.g. for comparison:

Operator	Description
`==`	Compares two objects for equality.
`!=`	Compares two objects for inequality.
`>`	True if the left hand side is greater than the right hand side.
`>=`	True if the left hand side is greater or equal to the right hand side.
`<`	True if the left hand side is lower than the right hand side.
`<=`	True if the left hand side is lower or equal to the right hand side.

Ansible uses Jinja2 tests and filters in conditionals. Ansible supports all the standard tests and filters, and adds some unique ones as well.

For more on this, please refer to the documentation: https://jinja.palletsprojects.com/en/latest/templates/

Step 1 - Install service conditionally

As an example we would like to install an FTP server, but only on hosts that are in the `ftpserver` inventory group.

To do that, first edit the inventory to add another group, and place `node2` in it. Make sure that that IP address of `node2` is always the same when `node2` is listed. Edit the inventory `~/lab_inventory/hosts` to look like the following listing:

```
[control]
ansible-1 ansible_connection=local

[web]
node1 ansible_host=node1.example.com
node2 ansible_host=node2.example.com
node3 ansible_host=node3.example.com

[ftpserver]
node2 ansible_host=node2.example.com
```

Next create the file `ftpserver.yml` on your control host in the `~/ansible_files/` directory:

```
---
- name: Install vsftpd if hosts are in inventory group
  hosts: all
  tasks:
    - name: Install FTP server when host in ftpserver group
      ansible.builtin.package:
        name: vsftpd
        state: present
      become: true
      when: inventory_hostname in groups["ftpserver"]
```

Tip

By now you should know how to run Ansible Playbooks, we'll start to be less verbose in this guide. Go create and run it. :-)

Run it and examine the output. The expected outcome: The task is skipped on node1, node3 and the ansible host (your control host) because they are not in the ftpserver group in your inventory file.

```
TASK [Install FTP server when host in ftpserver group]
*****************************************
skipping: [ansible-1]
skipping: [node1]
skipping: [node3]
changed: [node2]
```

In your condition the *magic variable* `inventory_hostname` is used, a variable set by Ansible itself. It contains the inventory name for the *current* host being iterated over in the play.

Step 2 - Find out exact version of installed service

We installed the *vsftpd* package, we would now be able to start the *vsftp*-Service (Very Secure FTP Daemon). But what version of the package is installed?

Lets add two more tasks to our playbook, one to gather information about all installed packages on the target host with the module *package_facts*. This module adds the gathered information to the `ansible_facts`, from there you can use the information as an Ansible variable. The last tasks outputs the exact version number to *stdout*, but only if the package is installed (the variable in the *packages* dictionary is defined).

```yaml
---
- name: Install vsftpd if hosts are in inventory group
  hosts: all
  tasks:
    - name: Install FTP server when host in ftpserver group
      ansible.builtin.package:
        name: vsftpd
        state: present
      become: true
      when: inventory_hostname in groups["ftpserver"]

    - name: Get information about installed packages
      ansible.builtin.package_facts:
        manager: auto

    - name: Debug exact version of installed vsFTP package
      ansible.builtin.debug:
        msg: "vsFTP is installed in Version {{ ansible_facts.packages.vsftpd.0.version }}"
      when: ansible_facts.packages.vsftpd is defined
```

As you can see, the simplest conditional statement applies to a single task. If you do this on your own, create the task, then add a `when` statement that applies a *test*. The `when` clause is a *raw* Jinja2 expression, you can use variables here, but you **don't** have to use double curly braces to enclose the variable.

> **Tip**
>
> Run the playbook and observe the output!

Looking back at the playbook, why did we add the condition to the last task?
We know that we installed the package, therefore the condition is always true (the variable is definitely defined) and we could live without the condition.

You should always strive towards making your playbooks as robust as possible, what would happen if we would change the first task to *de-install* the service and not use the condition?
Let's change the title and the *state* to `absent`, remove (or comment) the condition and run the playbook again.

```yaml
---
- name: Install vsftpd if hosts are in inventory group
  hosts: all
  tasks:
    - name: De-install FTP server when host in ftpserver group
      ansible.builtin.package:
        name: vsftpd
        state: absent
      become: true
      when: inventory_hostname in groups["ftpserver"]

    - name: Get information about installed packages
      ansible.builtin.package_facts:
        manager: auto

    - name: Debug exact version of installed vsFTP package
      ansible.builtin.debug:
        msg: "vsFTP is installed in Version {{ ansible_facts.packages.vsftpd.0.version }}"
    #  when: ansible_facts.packages.vsftpd is defined
```

The service is removed by the playbook, therefore the result of the *package_facts* module does **not** include the *vsftpd* package anymore (!) and your playbooks ends with an error message!

Step 3 - Challenge lab

Add a task to the playbook which outputs a message if important security patches are included in the installed *vsftpd* package.

- Make sure that the *vsftpd* package is installed (again).
- Output the message *"The version of vsftpd includes important security patches!"* if the version of *vsftpd* is **greater than 3.0**

> **Tip**
>
> The Ansible documentation is helpful, a *test* to compare versions is available.

> **Tip**
>
> You need to check multiple conditions this time (if the package is installed at all **and** if the version is greater than 3.0)! You can use *logical operators* (like `and`, `or`, `not`) to combine conditions. When you have multiple conditions that all need to be **true** (that is, a logical `and`), you can specify them as a **list**. Take a look at the documentation for additional information.

Run the extended playbook.

Solution

The updated playbook:

```
---
- name: Install vsftpd if hosts are in inventory group
  hosts: all
  tasks:
    - name: Install FTP server when host in ftpserver group
      ansible.builtin.package:
        name: vsftpd
        state: present
      become: true
      when: inventory_hostname in groups["ftpserver"]

    - name: Get information about installed packages
      ansible.builtin.package_facts:
        manager: auto

    - name: Debug exact version of installed vsFTP package
      ansible.builtin.debug:
        msg: "{{ ansible_facts.packages.vsftpd.0.version }}"
      when: ansible_facts.packages.vsftpd is defined

    - name: Output message when vsftpd version is greater than 3.0
      ansible.builtin.debug:
        msg: "The version of vsftp includes important security patches!"
      when:
        - ansible_facts.packages.vsftpd is defined
        - ansible_facts.packages.vsftpd.0.version is version('3.0', '>')
```

Running the playbook outputs the following:

```
...
TASK [Output message when vsftp version is greater than 3.0]
*****************************************************************
skipping: [node1]
skipping: [node3]
ok: [node2] => {
    "msg": "The version of vsftpd includes important security patches!"
}
...
```

The message is shown? Awesome!

To see if the conditions works, change the version to compare to `4.0` and run the playbook again. You should now see the last task as *skipped*.