Red Hat Enterprise Linux Automation with Ansible

Chapter 6. Managing Complex Plays and Playbooks ☆

Abstract

Goal	Write playbooks for larger, more complex plays and playbooks.
Objectives	• Write sophisticated host patterns to efficiently select hosts for a play or ad hoc command. Manage large playbooks by importing or including other playbooks or tasks from external files, either unconditionally or based on a conditional test.

Sections	• Selecting Hosts with Host Patterns (and Guided Exercise) • Including and Importing Files (and Guided Exercise)
Lab	• Managing Complex Plays and Playbooks

Selecting Hosts with Host Patterns

Objectives

After completing this section, you will be able to write sophisticated host patterns to efficiently select hosts for a play or ad hoc command.

Referencing Inventory Hosts

Host patterns are used to specify the hosts to target by a play or ad hoc command. In its simplest form, the name of a managed host or a host group in the inventory is a host pattern that specifies that host or host group.

You have already used host patterns in this course. In a play, the `hosts` directive specifies the managed hosts to run the play against. For an ad hoc command, provide the host pattern as a command line argument to the `ansible` command.

It is usually easier to control what hosts a play targets by carefully using host patterns and having appropriate inventory groups, instead of setting complex conditionals on the play's tasks. Therefore, it is important to have a robust understanding of host patterns.

The following example inventory is used throughout this section to illustrate host patterns.

```
[student@controlnode ~]$ cat myinventory
web.example.com
data.example.com

[lab]
labhost1.example.com
labhost2.example.com

[test]
test1.example.com
test2.example.com

[datacenter1]
labhost1.example.com
test1.example.com

[datacenter2]
labhost2.example.com
test2.example.com

[datacenter:children]
datacenter1
datacenter2

[new]
192.168.2.1
192.168.2.2
```

To demonstrate how host patterns are resolved, you will execute an Ansible Playbook, `playbook.yml`, using different host patterns to target different subsets of managed hosts from this example inventory.

Managed Hosts

The most basic host pattern is the name for a single managed host listed in the inventory. This specifies that the host will be the only one in the inventory that will be acted upon by the `ansible` command.

When the playbook runs, the first `Gathering Facts` task should run on all managed hosts that match the host pattern. A failure during this task can cause the managed host to be removed from the play.

If an IP address is listed explicitly in the inventory, instead of a host name, then it can be used as a host pattern. If the IP address is not listed in the inventory, then you cannot use it to specify the host even if the IP address resolves to that host name in the DNS.

The following example shows how a host pattern can be used to reference an IP address contained in an inventory.

```
[student@controlnode ~]$ cat playbook.yml
---
- hosts: 192.168.2.1
...output omitted...

[student@controlnode ~]$ ansible-playbook playbook.yml

PLAY [Test Host Patterns] ******************************************************

TASK [Gathering Facts] *********************************************************
ok: [192.168.2.1]
...output omitted...
```

NOTE

One problem with referring to managed hosts by IP address in the inventory is that it can be hard to remember which IP address matches which host for your plays and ad hoc commands. However, you may have to specify the host by IP address for connection purposes if the host does not have a resolvable host name.

It is possible to point an alias at a particular IP address in your inventory by setting the `ansible_host` host variable. For example, you could have a host in your inventory named `dummy.example`, and then direct connections using that name to the IP address 192.168.2.1 by creating a `host_vars/dummy.example` file containing the following host variable:

```
ansible_host: 192.168.2.1
```

Specifying Hosts Using a Group

You have already used inventory host groups as host patterns. When a group name is used as a host pattern, it specifies that Ansible will act on the hosts that are members of the group.

```
[student@controlnode ~]$ cat playbook.yml
---
- hosts: lab
...output omitted...
[student@controlnode ~]$ ansible-playbook playbook.yml

PLAY [Test Host Patterns] ******************************************************

TASK [Gathering Facts] *********************************************************
ok: [labhost1.example.com]
ok: [labhost2.example.com]
...output omitted...
```

Remember that there is a special group named `all` that matches all managed hosts in the inventory.

```
[student@controlnode ~]$ cat playbook.yml
---
- hosts: all
...output omitted...
[student@controlnode ~]$ ansible-playbook playbook.yml

PLAY [Test Host Patterns] ******************************************************

TASK [Gathering Facts] *********************************************************
ok: [labhost2.example.com]
ok: [test2.example.com]
ok: [web.example.com]
ok: [data.example.com]
ok: [labhost1.example.com]
ok: [192.168.2.1]
ok: [test1.example.com]
ok: [192.168.2.2]
```

There is also a special group named `ungrouped`, which includes all managed hosts in the inventory that are not members of any other group:

```
[student@controlnode ~]$ cat playbook.yml
---
- hosts: ungrouped
...output omitted...
[student@controlnode ~]$ ansible-playbook playbook.yml

PLAY [Test Host Patterns] ******************************************************

TASK [Gathering Facts] *********************************************************
ok: [web.example.com]
ok: [data.example.com]
```

Matching Multiple Hosts with Wildcards

Another method of accomplishing the same thing as the `all` host pattern is to use the asterisk (*) wildcard character, which matches any string. If the host pattern is just a quoted asterisk, then all hosts in the inventory will match.

```
[student@controlnode ~]$ cat playbook.yml
---
- hosts: '*'
...output omitted...
[student@controlnode ~]$ ansible-playbook playbook.yml

PLAY [Test Host Patterns] ******************************************************

TASK [Gathering Facts] *********************************************************
ok: [labhost2.example.com]
ok: [test2.example.com]
ok: [web.example.com]
ok: [data.example.com]
ok: [labhost1.example.com]
ok: [192.168.2.1]
ok: [test1.example.com]
ok: [192.168.2.2]
```

IMPORTANT

Some characters that are used in host patterns also have meaning for the shell. This can be a problem when using host patterns to run ad hoc commands from the command line with `ansible`. It is a recommended practice to enclose host patterns used on the command line in single quotes to protect them from unwanted shell expansion.

Likewise, if you are using any special wildcards or list characters in an Ansible Playbook, then you must put your host pattern in single quotes to ensure it is parsed correctly.

```
---
  hosts: '!test1.example.com,development'
```

The asterisk character can also be used to match any managed hosts or groups that contain a particular substring.

For example, the following wildcard host pattern matches all inventory names that end in `.example.com`:

```
[student@controlnode ~]$ cat playbook.yml
---
- hosts: '*.example.com'
...output omitted...
[student@controlnode ~]$ ansible-playbook playbook.yml

PLAY [Test Host Patterns] ******************************************************

TASK [Gathering Facts] *********************************************************
ok: [labhost1.example.com]
ok: [test1.example.com]
ok: [labhost2.example.com]
ok: [test2.example.com]
ok: [web.example.com]
ok: [data.example.com]
```

The following example uses a wildcard host pattern to match the names of hosts or host groups that start with `192.168.2.`:

```
[student@controlnode ~]$ cat playbook.yml
---
- hosts: '192.168.2.*'
...output omitted...
[student@controlnode ~]$ ansible-playbook playbook.yml

PLAY [Test Host Patterns] ******************************************************

TASK [Gathering Facts] *********************************************************
ok: [192.168.2.1]
ok: [192.168.2.2]
```

The next example uses a wildcard host pattern to match the names of hosts or host groups that begin with `datacenter`.

```
[student@controlnode ~]$ cat playbook.yml
---
- hosts: 'datacenter*'
...output omitted...
[student@controlnode ~]$ ansible-playbook playbook.yml

PLAY [Test Host Patterns] ************************************************

TASK [Gathering Facts] ***************************************************
ok: [labhost1.example.com]
ok: [test1.example.com]
ok: [labhost2.example.com]
ok: [test2.example.com]
```

IMPORTANT

The wildcard host patterns match all inventory names, hosts, and host groups. They do not distinguish between names that are DNS names, IP addresses, or groups, which can lead to some unexpected matches.

For example, compare the results of specifying the `datacenter*` host pattern from the preceding example with the results of the `data*` host pattern based on the example inventory:

```
[student@controlnode ~]$ cat playbook.yml
---
- hosts: 'data*'
...output omitted...
[student@controlnode ~]$ ansible-playbook playbook.yml

PLAY [Test Host Patterns] ************************************************

TASK [Gathering Facts] ***************************************************
ok: [labhost1.example.com]
ok: [test1.example.com]
ok: [labhost2.example.com]
ok: [test2.example.com]
ok: [data.example.com]
```

Lists

Multiple entries in an inventory can be referenced using logical lists. A comma-separated list of host patterns matches all hosts that match any of those host patterns.

If you provide a comma-separated list of managed hosts, then all those managed hosts will be targeted:

```
[student@controlnode ~]$ cat playbook.yml
---
- hosts: labhost1.example.com,test2.example.com,192.168.2.2
...output omitted...
[student@controlnode ~]$ ansible-playbook playbook.yml

PLAY [Test Host Patterns] ************************************************

TASK [Gathering Facts] ***************************************************
ok: [labhost1.example.com]
ok: [test2.example.com]
ok: [192.168.2.2]
```

If you provide a comma-separated list of groups, then all hosts in any of those groups will be targeted:

```
[student@controlnode ~]$ cat playbook.yml
---
- hosts: lab,datacenter1
...output omitted...
[student@controlnode ~]$ ansible-playbook playbook.yml

PLAY [Test Host Patterns] ************************************************

TASK [Gathering Facts] ***************************************************
ok: [labhost1.example.com]
ok: [labhost2.example.com]
ok: [test1.example.com]
```

You can also mix managed hosts, host groups, and wildcards, as shown below:

```
[student@controlnode ~]$ cat playbook.yml
---
- hosts: lab,data*,192.168.2.2
...output omitted...
[student@controlnode ~]$ ansible-playbook playbook.yml

PLAY [Test Host Patterns] ************************************************

TASK [Gathering Facts] ***************************************************
ok: [labhost1.example.com]
ok: [labhost2.example.com]
ok: [test1.example.com]
ok: [test2.example.com]
ok: [data.example.com]
ok: [192.168.2.2]
```

> **NOTE**
>
> The colon character (:) can be used instead of a comma. However, the comma is the preferred separator, especially when working with IPv6 addresses as managed host names. You may see the colon syntax in older examples.

If an item in a list starts with an ampersand character (&), then hosts must match that item in order to match the host pattern. It operates similarly to a logical AND.

For example, based on our example inventory, the following host pattern matches machines in the `lab` group only if they are also in the `datacenter1` group:

```
[student@controlnode ~]$ cat playbook.yml
---
- hosts: lab,&datacenter1
...output omitted...
[student@controlnode ~]$ ansible-playbook playbook.yml

PLAY [Test Host Patterns] ************************************************

TASK [Gathering Facts] ***************************************************
ok: [labhost1.example.com]
```

You could also specify that machines in the `datacenter1` group match only if they are in the `lab` group with the host patterns `&lab,datacenter1` or `datacenter1,&lab`.

You can exclude hosts that match a pattern from a list by using the exclamation point or "bang" character (!) in front of the host pattern. This operates like a logical NOT.

This example matches all hosts defined in the `datacenter` group, except `test2.example.com` based on the example inventory:

```
[student@controlnode ~]$ cat playbook.yml
---
- hosts: datacenter,!test2.example.com
...output omitted...
[student@controlnode ~]$ ansible-playbook playbook.yml

PLAY [Test Host Patterns] ******************************************************

TASK [Gathering Facts] *********************************************************
ok: [labhost1.example.com]
ok: [test1.example.com]
ok: [labhost2.example.com]
```

The pattern `'!test2.example.com,datacenter'` could have been used in the preceding example to achieve the same result.

The final example shows the use of a host pattern that matches all hosts in the test inventory, except the managed hosts in the `datacenter1` group.

```
[student@controlnode ~]$ cat playbook.yml
---
- hosts: all,!datacenter1
...output omitted...
[student@controlnode ~]$ ansible-playbook playbook.yml

PLAY [Test Host Patterns] ******************************************************

TASK [Gathering Facts] *********************************************************
ok: [web.example.com]
ok: [data.example.com]
ok: [labhost2.example.com]
ok: [test2.example.com]
ok: [192.168.2.1]
ok: [192.168.2.2]
```

REFERENCES

Working with Patterns — Ansible Documentation
(https://docs.ansible.com/ansible/2.9/user_guide/intro_patterns.html)

Working with Inventory — Ansible Documentation
(https://docs.ansible.com/ansible/2.9/user_guide/intro_inventory.html)

➜ NEXT (/ROL/APP/COURSES/RH294-8.4/PAGES/CH06S02)

All policies and guidelines (https://www.redhat.com/en/about/all-policies-guidelines)

Release Notes (https://learn.redhat.com/t5/Red-Hat-Learning-Subscription/Red-Hat-Learning-Subscription-Release-Notes/ba-p/22952)

Cookie Preferences