Ansible – PLAYBOOK YAML DeepDive

VISHWANATH M S

VISHWACLOUDLAB.COM

Topics

- Variables
  - Standard variables
  - Read a Variable
  - include_vars
  - VARS_FILES
  - set_fact
- Loops (Iteration)
- Conditionals
- Tags
- Blocks
- Handlers

Ansible – Variables

What is Variables?

- Variable lists are represented with **vars** key.
- Variables are either Sequence or maps

- Valid variables Name : Should be letters, numbers or underscore
- Example:

vars:

var1: “This is first variable”

Var_2: “This is second variable”

Variables usage

```
---
- hosts: web
  vars:
    my_cont: "this is test web server"

  tasks:
  - name: Install httpd on web server
    yum:
      name: httpd
      state: present
  - copy:
      dest: /var/www/html/index.html
      content: "{{ my_cont }}"
```

How to read a variable!!!

- With the help of **vars_promt** selection
- Syntax is:

vars_prompt:
  name: var1
  prompt: Enter the value !!!

Example:

```
---
- Name: Reading a variable
  hosts: localhost
  vars_prompt:
     name: var1
      prompt: Enter any value !!!
```

How to pass variable using cli!!!

Example: var-env.yml

```
---
- name: pass variable using cli
  hosts: localhost
  vars:
    var1: "{{ var11 }}"
  tasks:
   - debug: msg="Hello, {{ var1 }}!"
```

Run the below:

```
ansible-playbook var-env.yml -e "var11=world"
```

Include Variables from Another file

- The **include_vars** module

Example: include_vars

```
---
# ./vars/name_vars.yml

name: Everyone
```

Example:

```
---
# ./hello_world.yml

- name: print greeting
  hosts: "*"
  tasks:
    - include_vars: name_vars.yml

    - debug: msg="Hello, {{ name }}!"
```

vars_file Variables from Another file

- The **vars_file** keyword , works on only plays

Example: vars_file

```
---
# ./vars/my_vars.yml

foo: Everyone
```

Example: site.yml

```
---
- hosts: localhost
  vars_files:
    - "vars/{{ name }}.yml"
  tasks:
    - debug: var={{ foo }}
```

Run the below:

```
ansible-playbook site.yml -e "name=my_vars" -c local
```

Variables set with “set_fact”

- The **include_vars** module

Example:

```
---
# ./hello_world.yml

- name: print greeting
  hosts: "*"
  tasks:
    - set_fact: name=testing
    - include_vars: name_vars.yml

    - debug: msg="Hello, {{ name }}!"
```

Output:

```
ok: [10.10.10.10] => {
    "msg": "Hello, testing!"
}
```

Valid Variables name

YAML also supports dictionaries which map keys to values. For instance:

```
Example:
myvars:
  field1: personal
  field2: professional
```

```
We can then ref a specific field as
below
myvars['field1']
Myvars.field1
```

Variables in inventory

Example: inventory

```
[web-rhel]
host1
host2

[web-rhel:vars]
Apache_package: httpd
```

Cont …….

Cont …….

```
[web-ubuntu]
host3
host4

[web-ubuntu:vars]
Apache_package: apache2
```

```
$ ansible web-ubuntu -m apt -a 'name="{{ apache_package }}" state=latest'
$ ansible web-rhel -m yum -a 'name="{{ apache_package }}" state=latest'
```

Variables in inventory … Continued…

Example: **inventory in YAML**

```
web-rhel:
  hosts:
    host1:
      apache_package: httpd
      http_port: 80
      maxRequestsPerChild: 505
    host2:
      apache_package: apache2
      http_port: 80
      maxRequestsPerChild: 505
```

Ansible – LOOPS

What are loops and How to configure!!!

- Loops are also called as iteration
- Loops are used when there is an repeated activity to be performed again and again.

- Playbook supports Loops with **“with_items”**

- Example:
  To create three Directories

Why use Loops ??

Solution Without LOOPS

```
- name: This will create fold1
  command: mkdir /root/fold1
- name: This will create fold2
  command: mkdir /root/fold2
- name: This will create fold3
  command: mkdir /root/fold3
```

Why use Loops ??

Solution With LOOPS

```
---
- hosts: localhost
  name: To Create 3 Folders
  become: true
  tasks:
    command: mkdir /root/"{{ item }}"
    with_items:
    - fold1
    - fold2
    - fold3
```

Command line Variable:

We can access the command line variables in a playbook like a normal variables in the script.

```
---
- name: External Variable
  hosts: localhost
  tasks:
  - debug:
      msg: "The value of var1= {{ var1 }} and var2= {{ var2 }}"
```

To pass variables from command line to playbook?

$ ansible-playbook ext-var01.yml -e "var1=value var2=value"

Conditional Statements

Conditional statements are expressions and they work on Boolean values like true or false.

Based on conditional statements scripts can compute different actions.

Example:

We want to create a file in remote server.

If file exists then ?

Do not execute the action

Now, this is the condition, which needs to be checked before an action.

Conditional Statements Continues....

Syntax:

tasks:

- module: its relevant code

when: expression(result is always either true or false)

Example:

```
---
- name: install Apache Web-Server
  hosts: all
  tasks:
    - name: Install Apache on CentOS  Server
      yum: name=httpd  state=present
      become: yes
      when: ansible_os_family == "RedHat"

    - name: Install Apache on Ubuntu Server
      apt: name=apache2 state=present
      become: yes
      when: ansible_os_family == "Debian"
```

Conditional With Logical operator

AND operator

```
when: ansible_os_family == "Debian" and ansible_distribution_version == "18.04"
```

OR Operator

```
---
- name: Check disk space usage
  hosts: all
  tasks:
  - name: Check disk space usage on servers
    shell: df -Th
    register: result
  - debug:
      var: result.stdout_lines
    when: ansible_os_family == "Debian" or ansible_os_family == "RedHat"
```

Questions……