



ANSIBLE

# ANSIBLE BASIC LAB MANUAL

Student Lab Kit v1.1

## ABSTRACT

This lab manual is designed for students who are interested in Ansible Basic Automation

**Confidential Document**

Working with Templates

## Table of Contents

<b>Lab Overview and objectives</b>	<b>2</b>
<i>Guided Tasks</i>	2
Template Module	2
Task 1: Template module – basic usage	2
Task 2: Customize webservers home page using a template	3
Task 3: Using “for” in Jija2 template	4
Jinja filters	5
Task 4: Number filters (int, abs, round)	5
Task 5: Forcing variables to be defined	6
Task 6: Defaulting variables	7
Task 7: List filters	7
Task 8: Random and Shuffle filters	9
Task 9: IP Address filters	10
Task 10: Hashing filters	10

# Lab Overview and objectives

The purpose of this lab is to learn how to use Jinja2 templates in Ansible in order to apply custom modifications on the content of your files. For simple modifications **lineinfile** and **blockinfile** modules can be used, but for advanced modifications, which contains also variables and facts using Jinja2 templates is what we need.

## Guided Tasks

### Template Module

#### Task 1: Template module – basic usage

In order to use Jinja2 templates in Ansible to perform modifications in files we have to use **template** module, which has mainly two parameters: **src** and **dest**, exactly like **copy** module, because the usage is similar somehow, except that the **src** file contains references to variables or facts. Let's make a simple example to see how templating works. We are going to create first the template file (**src** file), which usually has the **.j2** extension just to know that is a Jinja2 template:

```
student@ansible-00-01-hivemaster:~$ vi template.j2
Nickname: {{ nickname }}
Email Address: {{ email }}
Description: {{ description }}
Role: {{ role }}
Organization: {{ org }}
```

Next, we create the playbook:

```
student@ansible-00-01-hivemaster:~$ vi templating.yml
---
- hosts: hivemaster
  become: yes
  gather_facts: no
  vars:
    nickname: bogdan
    email: bogdan@sass.ro
    description: IT guy
    role: admin
    org: SASS Training
  tasks:
    - name: deploy file from template
      template:
        src: template.j2
```

```
dest: /tmp/info.txt
```

Run the playbook (`ansible-playbook templating.yml`) and explore the results in `/tmp/info.txt`:

```
student@ansible-00-01-hivemaster:~$ cat /tmp/info.txt
Nickname: bogdan
Email Address: bogdan@sass.ro
Description: IT guy
Role: admin
Organization: SASS Training
```

## Task 2: Customize webserver's home page using a template

In the previous lab we installed Apache on our hosts from `webserver` group. It's time to customize index page for each host, starting from a Jinja2 template. Let's create the template first:

```
student@ansible-00-01-hivemaster:~$ vi index.html.j2
<html>
<head>
<title>Welcome to Ansible</title>
</head>
<body>
<p>
Server details:
Hostname: {{ ansible_hostname }}
OS: {{ ansible_distribution }} {{ ansible_distribution_version }}
</p>
</body>
</html>
```

Now let's create a playbook to deploy this template to webserver's:

```
student@ansible-00-01-hivemaster:~$ vi customize_homepage.yml
---
- name: Customize homepage
  hosts: webserver
  become: yes
  tasks:
    - name: Deploy file from j2 template
      template:
        src: index.html.j2
        dest: /var/www/html/index.html
```

Run the playbook and then test the results using `curl`:

```
student@ansible-00-01-hivemaster:~$ curl ansible-00-02-ubuntu
<html>
```

```
<head>
<title>Welcome to Ansible</title>
</head>
<body>
<p>
Server details:
Hostname: ansible-00-02-ubuntu
OS: Ubuntu 18.04
</p>
</body>
</html>
```

```
student@ansible-00-01-hivemaster:~$ curl ansible-00-03-centos
```

```
<html>
<head>
<title>Welcome to Ansible</title>
</head>
<body>
<p>
Server details:
Hostname: ansible-00-03-centos
OS: CentOS 7.7
</p>
</body>
</html>
```

At this point you may encounter an error trying to access the webserver from centos server. This is because the centos server has the firewall active. The default firewall software on centos is firewalld. So, make sure that you add a rule or stop the firewall in order to be able to access the webpage. Here is an example on how you can disable firewalld using ansible:

```
---
- name: Disable firewall
  hosts: webservers
  become: yes
  tasks:
    - name: Disable and stop firewalld
      service:
        name: firewalld
        enabled: no
        state: stopped
      when: ansible_facts['os_family'] == "RedHat"
```

### Task 3: Using “for” in Jija2 template

There may be some tasks when you need to iterate through a list of variables in a Jinja2 template. In order to do this we have to use some special delimiters `{% ... %}`:

```
student@ansible-00-01-hivemaster:~$ vi iterate.j2
```

```
{% for item in my_list %}
  {{ item }}
{% endfor %}
```

```
student@ansible-00-01-hivemaster:~$ vi for_in_jinja.yml
---
- name: For loop in J2
  hosts: hivemaster
  gather_facts: no
  vars:
    my_list: [Ford, Audi, BMW, Skoda, Jaguar, Citroen]
  tasks:
    - name: Deploy file from template
      template:
        src: iterate.j2
        dest: /tmp/file.txt
        mode: 0644
```

Now let's cat the content of file.txt:

```
student@ansible-00-01-hivemaster:~$ cat /tmp/file.txt
Ford
Audi
BMW
Skoda
Jaguar
Citroen
```

You can read more about Jinja2 Templates at <https://jinja.palletsprojects.com/en/2.10.x/templates/>

## Jinja2 filters

Jinja2 filters are a way to transform expressions from one type of data to another. The syntax to apply Jinja2 filters to template variables is the vertical bar character |, also called a 'pipe' in Unix environments (ex: `{{variable|filter}}`). It's worth mentioning you can apply multiple filters to the same variable (ex: `{{variable|filter|filter}}`).

### Task 4: Number filters (int, abs, round)

During “Facts and Variables” lab we used extra variables for Task 7 and you noticed that instead of calculating the correct sum of the 2 vars, Ansible returned the concatenation of the 2 values as strings. We are going to fix that right now using Jinja2 `int` filter:

```
student@ansible-00-01-hivemaster:~$ vi sum.yml
---
- name: Sum
  hosts: hivemaster
  gather_facts: no
```

```
vars:
  var1: 2
  var2: 3
tasks:
- name: Print sum of vars
  debug:
    msg: "{{ var1|int + var2|int }}"
```

```
student@ansible-00-01-hivemaster:~$ ansible-playbook sum.yml -e
"var1=20" -e "var2=30"
```

```
PLAY [Sum]
*****
TASK [Print sum of vars]
*****
ok: [hivemaster] => {
  "msg": "50"
}
```

Let's try to apply `abs` (absolute value) filter over our sum, made now from an integer and a negative float and rounded number:

```
---
- name: Sum
  hosts: hivemaster
  gather_facts: no
  vars:
    var1: 2
    var2: 3
  tasks:
- name: Print sum of vars
  debug:
    msg: "{{ (var1|int + var2|float|round)|abs }}"
```

```
student@ansible-00-01-hivemaster:~$ ansible-playbook sum.yml -e
"var1=20" -e "var2=-30.33"
```

```
TASK [Print sum of vars]
*****
ok: [hivemaster] => {
  "msg": "10.0"
}
```

## Task 5: Forcing variables to be defined

By default Ansible fails if a variable is not defined, but this can be disabled in `ansible.cfg` by uncommenting line `error_on_undefined_vars = False`. After that, we can explicitly specify that a variable should be defined using `{{ variable | mandatory }}`. Add this task to previous playbook:

```
- name: Mandatory var
  debug:
    msg: "{{ var3 | mandatory }}"
```

```
student@ansible-00-01-hivemaster:~$ ansible-playbook sum.yml -e
"var1=20" -e "var2=-30.33"
```

```
PLAY [Sum]
*****

TASK [Print sum of vars]
*****
ok: [hivemaster] => {
  "msg": "10.0"
}

TASK [Mandatory var]
*****
fatal: [hivemaster]: FAILED! => {"msg": "Mandatory variable 'var3' not
defined."}
```

## Task 6: Defaulting variables

Using the default filter we can specify a default value for undefined variables. Modify previous task:

```
- name: Mandatory var -> defaulting var
  debug:
    msg: "{{ var3 | default(50) }}"
```

## Task 7: List filters

**min, max filters:**

```
student@ansible-00-01-hivemaster:~$ vi list_filters.yml
---
- name: List filters Playbook
  hosts: hivemaster
  gather_facts: no
  vars:
    list1:
      - 77
      - 50
      - 100
      - -400
      - 100
    list2:
      - 400
      - 77
```



```
- 300
- 100
- 77

tasks:
- name: Get minimum value of list
  debug:
    msg: "{{ list1|min }}"
```

```
student@ansible-00-01-hivemaster:~$ ansible-playbook list_filters.yml
TASK [Get minimum value of list]
*****
ok: [hivemaster] => {
  "msg": "-400"
}
```

`unique`, `union`, `intersect` filters:

```
- name: Get unique values of list
  debug:
    msg: "{{ list1|unique }}"
```

```
TASK [Get unique values of list]
*****
ok: [hivemaster] => {
  "msg": [
    77,
    50,
    100,
    -400
  ]
}
```

```
- name: Get union of two lists
  debug:
    msg: "{{ list1|union(list2) }}"
```

```
TASK [Get union of two lists]
*****
ok: [hivemaster] => {
  "msg": [
    77,
    50,
    100,
    -400,
    400,
    300
  ]
}
```

```
- name: Get intersection of two lists
  debug:
    msg: "{{ list1|intersect(list2) }}"
```

```
TASK [Get intersection of two lists]
*****
ok: [hivemaster] => {
  "msg": [
    77,
    100
  ]
}
```

## Task 8: Random and Shuffle filters

Add these tasks to the same playbook (list\_filters.yml):

```
- name: Get random number from a list
  debug:
    msg: "{{ (list1|intersect(list2))|random }}"
```

```
TASK [Get random number from a list]
*****
ok: [hivemaster] => {
  "msg": "100"
}
```

```
- name: Get random number with start and step
  debug:
    msg: "{{ 666|random(start=10, step=7) }}"
```

```
TASK [Get random number with start and step]
*****
ok: [hivemaster] => {
  "msg": "206"
}
```

```
- name: Get a list shuffled
  debug:
    msg: "{{ (list1|union(list2))|shuffle }}"
```

```
TASK [Get a list shuffled]
*****
ok: [hivemaster] => {
  "msg": [
    77,
    400,
    300,
    100,
```

```
        50,  
        -400  
    ]  
}
```

### Task 9: IP Address filters

This is a useful filter to test if a string is a valid IP address. Notice that you have to install `netaddr` for this filter to work:

```
sudo apt-get install -y python-netaddr
```

```
- name: Test string for IP Addr  
  debug:  
    msg: "{{ '192.168.100.260'|ipaddr }}"
```

```
TASK [Test string for IP Addr]  
*****  
ok: [hivemaster] => {  
    "msg": false  
}
```

`ipv4`, `ipv6` can also be used to test specific protocol IP addresses.

### Task 10: Hashing filters

We already used hash filter during a previous lab when we set the password for a Linux user that we created.

```
student@ansible-00-01-hivemaster:~$ vi password.yml  
---  
- name: User management  
  hosts: all  
  become: yes  
  vars:  
    user_pass: 123abc  
  tasks:  
    - name:  
      user:  
        name: testuser  
        password: "{{ user_pass | password_hash('sha512') }}"
```

Remember that this task always returned “changed” even if we set the same password (because the password is salted in Linux). We can set a default `salt` for this task, but it is recommended to make it particular at least for each host (not to use the same salt for the entire infrastructure). So we will use `random` filter with a host-specific seed:

```
password:  "{{ user_pass | password_hash('sha512', 65534 |
random(seed=inventory_hostname) | string) }}"
```

There are also other types of hashes available: `md5`, `checksum`, `blowfish` etc. Go back to the previous yml file and add the following tasks:

```
- name: Get md5 hash
  debug:
    msg: "{{ list1.0 | hash('md5') }}"
```

```
TASK [Get md5 hash]
*****
ok: [hivemaster] => {
  "msg": "28dd2c7955ce926456240b2ff0100bde"
}
```

## Task 11: Other useful filters

Using `quote` filter we can add quotes to a variable (useful for shell usage):

```
- name: Add quotes
  debug:
    msg: "{{ item | quote }}"
  loop: "{{ list1 }}"
```

```
TASK [Add quotes for shell usage]
*****
ok: [hivemaster] => (item=77) => {
  "msg": "77"
}
ok: [hivemaster] => (item=50) => {
  "msg": "50"
}
ok: [hivemaster] => (item=100) => {
  "msg": "100"
}
ok: [hivemaster] => (item=-400) => {
  "msg": "-400"
}
ok: [hivemaster] => (item=100) => {
  "msg": "100"
}
```

Concatenate a list into a string:

```
- name: Concatenate list into string
  debug:
    msg: "{{ list1 | join(' ') }}"
```

```
TASK [Concatenate list into string]
*****
ok: [hivemaster] => {
  "msg": "77 50 100 -400 100"
}
```

Get the file name (basename) from a path:

```
- name: Get basename of path
  debug:
    msg: "{{ '/etc/ansible/ansible.cfg' | basename }}"
```

```
TASK [Get basename of path]
*****
ok: [hivemaster] => {
  "msg": "ansible.cfg"
}
```