



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**

Facts. Variables.

## Table of Contents

<b>Lab Overview and objectives</b>	<b>2</b>
<i>Guided Tasks</i>	2
Task 1: Fact gathering	2
Task 2: Disable fact gathering in Playbooks	2
Task 3: Custom facts	3
Task 4: Host and group variables	4
Task 5: Defining variables	5
Task 6: Including variables	6
Task 6: Extra variables	7
Task 6: Variables vs Facts	8

# Lab Overview and objectives

The purpose of this lab is to learn how to use facts and variables in playbooks. Facts are automatically gathered using “setup” module and they represent useful variables about remote hosts that can be used in playbooks. Variables are defined by user and not gathered by ansible, but they have basically the same purpose – making easier the process of managing different hosts.

## Guided Tasks

### Task 1: Fact gathering

We already tested the “setup” module even from the first labs, using Ansible ad-hoc command:

```
student@ansible-00-01-hivemaster:~$ ansible all -m setup -a
"filter=*ipv4"
hivemaster | SUCCESS => {
  "ansible_facts": {
    "ansible_default_ipv4": {
      "address": "10.128.0.48",
      "alias": "ens4",
      "broadcast": "global",
      "gateway": "10.128.0.1",
      "interface": "ens4",
      "macaddress": "42:01:0a:80:00:30",
      "mtu": 1460,
      "netmask": "255.255.255.255",
      "network": "10.128.0.48",
      "type": "ether"
    },
    "discovered_interpreter_python": "/usr/bin/python3"
  },
  "changed": false
}
[...]
```

### Task 2: Disable fact gathering in Playbooks

As you have already seen, at the beginning of the execution of a playbook, the first “implicit” task is `gather_facts`. This task can be disabled if we don’t need these variables collected about hosts, as this task is taking some additional time:

```
vi disable_facts.yml
---
```

```
- name: Disable facts
  hosts: all
  become: yes
  gather_facts: no
  tasks:
    - name: Print message
      debug:
        msg: "Fact gathering is disabled. Playbook is running faster"
```

```
student@ansible-00-01-hivemaster:~$ ansible-playbook disable_facts.yml
```

```
PLAY [Disable facts]
*****
TASK [Print message]
*****
ok: [ubuntu] => {
  "msg": "Fact gathering is disabled. Playbook is running faster"
}
[...]
```

### Task 3: Custom facts

Apart from the facts collected by Ansible from remote hosts, we can also add custom facts that Ansible can fetch from these hosts. To add custom facts we have to create a new directory “/etc/ansible/facts.d” and put inside it one (or more) \*.fact file (s), that return JSON formatted data, which will then be included in the raft of facts that Ansible gathers at the start of each playbook run.

We are going to create a shell script which returns some info about the current logged-in user and his IP address in JSON format (we are also going to make this file executable):

```
student@ansible-00-01-hivemaster:~$ sudo mkdir /etc/ansible/facts.d
student@ansible-00-01-hivemaster:~$ sudo vi
/etc/ansible/facts.d/test.fact

#!/bin/bash
user=$(who | cut -f 1 -d " " | tail -1 )
ipuser=$(who -u | cut -f 2 -d "(" | cut -f 1 -d ")" | head -1)

cat << EOF
{
  "user" : "$user",
  "ipuser" : "$ipuser"
}
student@ansible-00-01-hivemaster:~$ sudo chmod +x
/etc/ansible/facts.d/test.fact

student@ansible-00-01-hivemaster:~$ ansible hivemaster -m setup | grep
ipuser
"ipuser": "92.137.23.156",
```

## Task 4: Host and group variables

Before starting to use variables it is important to know that variable names should contain only letters, numbers and underscores, and should always start with a letter.

We already defined variables in inventory files (remember `ansible_host`, `ansible_user` etc) and we are going to access them in a playbook:

```
student@ansible-00-01-hivemaster:~$ vi print_vars.yml
---
- name: Host and group vars
  hosts: all
  become: true
  gather_facts: no
  tasks:
    - name: Print ansible_hostname
      debug:
        var: ansible_host

    - name: Print ansible_user
      debug:
        var: ansible_user

    - name: Print ansible_ssh_private_key_file
      debug:
        var: ansible_ssh_private_key_file
```

Notice that we disabled gathering facts! Run the playbook and explore the values of variables:

```
student@ansible-00-01-hivemaster:~$ ansible-playbook print_vars.yml

PLAY [Vars from inventory]
*****
TASK [Print ansible_hostname]
*****
ok: [ubuntu] => {
  "ansible_host": "10.128.0.49"
}
ok: [centos] => {
  "ansible_host": "10.142.15.213"
}
ok: [hivemaster] => {
  "ansible_host": "10.128.0.48"
}

TASK [Print ansible_user]
*****
ok: [ubuntu] => {
  "ansible_user": "ansible"
}
ok: [centos] => {
  "ansible_user": "ansible"
```

```

}
ok: [hivemaster] => {
  "ansible_user": "ansible"
}

TASK [Print ansible_ssh_private_key_file]
*****
ok: [ubuntu] => {
  "ansible_ssh_private_key_file": "/home/student/ansible_key"
}
ok: [centos] => {
  "ansible_ssh_private_key_file": "/home/student/ansible_key"
}
ok: [hivemaster] => {
  "ansible_ssh_private_key_file": "/home/student/ansible_key"
}

```

We also defined in the inventory file some variables (`apache_port` and `apache_path`) just for `webserver` group (ubuntu and centos), so let's add 2 more debug tasks:

```

- name: Print apache_port
  debug:
    var: apache_port

- name: Print apache_path
  debug:
    var: apache_path

```

Running the playbook you will notice that for our hivemaster server these 2 variables will be listed as “NOT DEFINED”, because this host is not inside the webserver group. We can also change the targeted hosts of this playbook to affect only webserver group:

```

---
- name: Host and group vars
  hosts: webserver
  become: true
  gather_facts: no

```

## Task 5: Defining variables

To begin with, we are going to define some variables and assign some values from facts to those variables:

```

student@ansible-00-01-hivemaster:~$ vi variables.yml
---
- name: Facts - Variables
  hosts: all
  become: true
  vars:
    - ubuntu_ip: "{{ hostvars['ubuntu']['ansible_default_ipv4']['address'] }}"
    - ubuntu_hostname: "{{ hostvars['ubuntu']['ansible_hostname'] }}"

```

```
- centos_ip: "{{ hostvars['centos']['ansible_default_ipv4']['address'] }}"
- centos_hostname: "{{ hostvars['centos']['ansible_hostname'] }}"
- hivemaster_ip: "{{
hostvars['hivemaster']['ansible_default_ipv4']['address'] }}"
- hivemaster_hostname: "{{ hostvars['hivemaster']['ansible_hostname'] }}"

tasks:
- name: Print ansible hostname
  debug:
    msg: "{{ ansible_hostname }}"

- name: Print IP address
  debug:
    msg: "{{ ansible_default_ipv4.address }}"

- name: Print inventory hostname
  debug:
    msg: "{{ inventory_hostname }}"
```

Running the playbook will print the information about defined variables, populated with values from facts:

```
student@ansible-00-01-hivemaster:~$ ansible-playbook variables.yml

PLAY [Facts - Variables]
*****
TASK [Gathering Facts]
*****
ok: [ubuntu]
ok: [hivemaster]
ok: [centos]

TASK [Print ansible hostname]
*****
ok: [ubuntu] => {
  "msg": "ansible-00-02-ubuntu"
}
ok: [centos] => {
  "msg": "ansible-00-03-centos"
}
ok: [hivemaster] => {
  "msg": "ansible-00-01-hivemaster"
}
[...]
```

## Task 6: Including variables

We can also include variables from other files using `include_vars` directive. Let's create first 2 var files:

```
student@ansible-00-01-hivemaster:~$ vi v1.yml
---
variable1: "Hello "
```

```
student@ansible-00-01-hivemaster:~$ vi v2.yml
---
variable2: "WORLD"
```

Now create a playbook and include these 2 files:

```
student@ansible-00-01-hivemaster:~$ vi include_vars.yml
---
- name: Including vars
  hosts: hivemaster
  gather_facts: no
  tasks:
    - name: include v1
      include_vars: "/home/student/v1.yml"

    - name: include v2
      include_vars: "/home/student/v2.yml"

- debug:
    msg: "{{ variable1 + variable2 }}"
```

Run the playbook:

```
student@ansible-00-01-hivemaster:~$ ansible-playbook include_vars.yml
[...]
TASK [debug]
*****
ok: [hivemaster] => {
    "msg": "Hello WORLD"
}
[...]
```

You are going to learn in Ansible Vault lab how to include more files at once in a playbook.

### Task 7: Extra variables

Firstly, let's create a simple playbook with a basic task (return the sum of 2 vars):

```
student@ansible-00-01-hivemaster:~$ vi sum.yml
---
- name: Sum of 2 vars
  hosts: hivemaster
  gather_facts: no
  vars:
    var1: 2
    var2: 3
  tasks:
    - name: Print sum of vars
```



```
debug:
  msg: "{{ var1 + var2 }}"
```

Basically, this task just performs the sum of 2 vars:

```
student@ansible-00-01-hivemaster:~$ ansible-playbook sum.yml

PLAY [Sum]
*****
TASK [Print sum of vars]
*****
ok: [hivemaster] => {
  "msg": "5"
}
```

Right now we are going to pass values for `var1` and `var2` from command line while running the playbook as extra vars:

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

You probably expect that the new values for our variables will replace the older ones (defined in the playbook) as the extra vars has precedence, but what you (probably) don't expect is that extra vars are treated like "strings" and the output is concatenation of these 2 strings:

```
TASK [Print sum of vars]
*****
ok: [hivemaster] => {
  "msg": "2030"
}
```

Notice that we are going to fix this behavior during "Jinja filters" section in the Templates lab!

## Task 8: Variables vs Facts

There is a main difference between variables and facts and we are going to see it by creating 2 separate playbooks and compare them:

```
student@ansible-00-01-hivemaster:~$ vi vars_vs_facts_1.yml
---
- name: Facts vs Vars
  hosts: hivemaster
  vars:
    var_time: "Var: {{lookup('pipe', 'date \"+%H:%M:%S\"')}}"
```

```
tasks:
  - name: Print var_time first time
    debug:
      msg: "{{ var_time }}"
```

```
- name: Sleep a little bit
  pause:
    seconds: 5

- name: Print var_time second time
  debug:
    msg: "{{ var_time }}"
```

Running this playbook will result in printing the `var_time` variable 2 times with a 5 seconds delay. Each time the variable is evaluated:

```
student@ansible-00-01-hivemaster:~$ ansible-playbook vars_vs_facts_1.yml

PLAY [Facts vs Vars]
*****
TASK [Gathering Facts]
*****
ok: [hivemaster]

TASK [Print var_time first time]
*****
ok: [hivemaster] => {
  "msg": "Var: 19:09:29"
}

TASK [Sleep a little bit]
*****
Pausing for 5 seconds
(ctrl+C then 'C' = continue early, ctrl+C then 'A' = abort)
ok: [hivemaster]

TASK [Print var_time second time]
*****
ok: [hivemaster] => {
  "msg": "Var: 19:09:34"
}
```

In the second playbook we are going to set a fact instead of a variable:

```
student@ansible-00-01-hivemaster:~$ vi vars_vs_facts_2.yml
---
- name: Facts vs Vars
  hosts: hivemaster
  tasks:
    - name: Set fact task
      set_fact:
        fact_time: "{{lookup('pipe', 'date \"+%H:%M:%S\"')}}"
    - name: Print fact_time first time
      debug:
        msg: "{{ fact_time }}"

    - name: Sleep a little bit
```

```
    pause:
      seconds: 5

- name: Print fact_time second time
  debug:
    msg: "{{ fact_time }}"
```

Running this playbook you will notice that the same value is printed both times (which means that the fact is set once during the execution and its value is not changed anymore):

```
student@ansible-00-01-hivemaster:~$ ansible-playbook vars_vs_facts_2.yml

PLAY [Facts vs Vars]
*****

TASK [Gathering Facts]
*****
ok: [hivemaster]

TASK [Set fact task]
*****
ok: [hivemaster]

TASK [Print fact_time first time]
*****
ok: [hivemaster] => {
  "msg": "19:11:25"
}

TASK [Sleep a little bit]
*****
Pausing for 5 seconds
(ctrl+C then 'C' = continue early, ctrl+C then 'A' = abort)
ok: [hivemaster]

TASK [Print fact_time second time]
*****
ok: [hivemaster] => {
  "msg": "19:11:25"
}
```