



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**

Running Ad-Hoc Commands

## Table of Contents

<b>Lab Overview and objectives</b>	<b>2</b>
<i>Guided Tasks</i>	2
Task 1: Testing Ad-Hoc commands on localhost	2
Task 2: Create a basic inventory file	3
Task 3: Running Ad-Hoc commands using password authentication	4
Task 3.1: Ping all hosts – Attempt 1	4
Task 3.2: Ping all hosts – Attempt 2	4
Task 3.3: Fix [DEPRECATION WARNING] for hive master	5
Task 3.4: Test again for hive master	5
Task 3.5: Check the connection user used by Ansible	6
Task 3.6: Accessing system files (privilege escalation)	6
Task 4: Running Ad-Hoc commands using key-based authentication	7
Task 4.1: Generate a SSH keypair	8
Task 4.2: Put publickey into correct place	9
Task 4.3: Ping all hosts (using key-based auth)	10
Task 4.4: Accessing system files (privilege escalation)	10
Task 4.5: Command module	11
Task 4: Setup module in Ad-Hoc commands	11

# Lab Overview and objectives

The purpose of this lab is testing Ansible connection to remote servers using different authentication methods by running Ad-Hoc commands.

Ad-Hoc commands are one line commands that performs one task on the target host. Using Ad-Hoc commands we can explore Ansible itself, without creating a playbook first, but having the opportunity to perform several tasks, such as managing services, rebooting servers, editing configurations, copying files, installing packages and so on.

Basic structure of an Ad-Hoc command:

```
$ ansible <pattern> [options] [module options]
```

Using **patterns** we choose which managed hosts or groups we want to execute against. A pattern can refer to a single host, IP, an inventory group, a set of groups or even all hosts of inventory.

Here are some options available for ansible command:

Option	Description
-i	inventory file (default /etc/ansible/hosts) or comma-separated list
-v	verbose mode (-vvv for more, -vvvv for connection debugging)
-m	module name to execute (default=command)
-a	module arguments
-b	privilege escalation ("become") method to use (default=sudo)
-C	("check") - don't make any changes; instead try to predict some of the changes
-f	number of parallel processes (forks) to use (default=5).

## Guided Tasks

### Task 1: Testing Ad-Hoc commands on localhost

In order to start managing servers we have to tell Ansible what hosts it can manage. The default inventory file is located in `/etc/ansible/hosts` but this file is empty (commented) by default. Before writing

our first ‘lite’ inventory file, we are going to test Ansible on `localhost`, which is “implicit”, so we don’t have to create an inventory file for it.

Let’s run `ansible` command with ‘localhost’ as <pattern> and using ping module:

```
student@ansible-00-01-hivemaster:~$ ansible localhost -m ping
localhost | SUCCESS => {
  "changed": false,
  "ping": "pong"
}
```

By default, Ansible is trying to run commands as the current user (unless another user is specified). We can see this running “id” command using “raw” module (Ansible should run under `student` user):

```
student@ansible-00-01-hivemaster:~$ ansible localhost -m raw -a "id"
localhost | CHANGED | rc=0 >>
uid=1002(student) gid=1003(student) groups=1003(student)
```

Remember that “raw” module is a little bit different than other modules, as it does not requires that Python should be installed on remote host, so it’s also the tool to perform Python installation in case that it is missing from the (current state of) distribution.

## Task 2: Create a basic inventory file

As we mentioned earlier, `ansible` command requires a <pattern>, to match some specific hosts from the inventory file. For the moment, we are going to create a ‘lite’ inventory file, which should contain only hostnames (or IPs) of our servers. You can find the details of hosts from your own pod in `/etc/hosts`:

```
student@ansible-00-01-hivemaster:~$ cat /etc/hosts
127.0.0.1      localhost
127.0.1.1      ansible-00-01-hivemaster.training.sass.ro ansible-00-01-
hivemaster

# The following lines are desirable for IPv6 capable hosts
::1           localhost ip6-localhost ip6-loopback
ff02::1       ip6-allnodes
ff02::2       ip6-allrouters

10.128.0.48    ansible-00-01-hivemaster.training.sass.ro ansible-00-01-
hivemaster ansible-00-01 srv01
10.128.0.49    ansible-00-02-ubuntu.training.sass.ro ansible-00-02-ubuntu
ansible-00-02 srv02
10.142.15.213  ansible-00-03-centos.training.sass.ro ansible-00-03-centos
ansible-00-03 srv03
```

Let’s create the inventory:

```
student@ansible-00-01-hivemaster:~$ vi hosts
```

```
ansible-00-01-hivemaster
ansible-00-02-ubuntu
10.142.15.213
```

Notice that I added for `hivemaster` and `ubuntu` their hostnames and for `centos` its IP address (just to show you that we can use any of them). We didn't specified any groups for our hosts, but Ansible creates by default `all` group, which contains all the hosts in the inventory.

### Task 3: Running Ad-Hoc commands using password authentication

#### Task 3.1: Ping all hosts – Attempt 1

Now that we have an inventory file, we can start running ad-hoc commands, so let's ping all hosts in the inventory:

```
student@ansible-00-01-hivemaster:~$ ansible -i hosts all -m ping
ansible-00-01-hivemaster | UNREACHABLE! => {
  "changed": false,
  "msg": "Failed to connect to the host via ssh: student@ansible-00-01-hivemaster: Permission denied (publickey,password).",
  "unreachable": true
}
ansible-00-02-ubuntu | UNREACHABLE! => {
  "changed": false,
  "msg": "Failed to connect to the host via ssh: student@ansible-00-02-ubuntu: Permission denied (publickey,password).",
  "unreachable": true
}
10.142.15.213 | UNREACHABLE! => {
  "changed": false,
  "msg": "Failed to connect to the host via ssh: student@10.142.15.213: Permission denied (publickey,gssapi-keyex,gssapi-with-mic,password).",
  "unreachable": true
}
```

#### Task 3.2: Ping all hosts – Attempt 2

You will notice that Ansible failed to connect to any of the hosts in the inventory. As we mentioned in the first task, Ansible is trying to connect using current user (student), but you can see that it failed for all hosts, because we didn't provided the password for student account. So, the proper way to run ansible using password authentication is use the options `-u` for username and `--ask-pass` for prompting us to enter the account password:

```
student@ansible-00-01-hivemaster:~$ ansible -i hosts all -m ping -u student --ask-pass
SSH password:
ansible-00-02-ubuntu | SUCCESS => {
  "ansible_facts": {
```

```

        "discovered_interpreter_python": "/usr/bin/python3"
    },
    "changed": false,
    "ping": "pong"
}
[DEPRECATION WARNING]: Distribution Ubuntu 18.04 on host ansible-00-01-
hivemaster should use /usr/bin/python3, but is using /usr/bin/python for
backward compatibility with prior Ansible releases. A future Ansible
release will default to using the discovered platform python for this
host. See
https://docs.ansible.com/ansible/2.9/reference_appendices/interpreter_di
scovery.html for more information. This feature will be
removed in version 2.12. Deprecation warnings can be disabled by setting
deprecation_warnings=False in ansible.cfg.
ansible-00-01-hivemaster | SUCCESS => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python"
    },
    "changed": false,
    "ping": "pong"
}
10.142.15.213 | SUCCESS => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python"
    },
    "changed": false,
    "ping": "pong"
}

```

### Task 3.3: Fix [DEPRECATION WARNING] for hivemaster

Notice that in this case we can omit to specify `-u` (for user) because “student” user is the current user that Ansible is trying to use.

Moreover, you can see that for our `hivemaster` server we have a [DEPRECATION WARNING], so we are going to fix this warning, as it is always recommended to do with warnings by adding `interpreter_python = auto` in the defaults section:

```

student@ansible-00-01-hivemaster:~$ sudo vi /etc/ansible/ansible.cfg

[defaults]

# some basic default values...
interpreter_python = auto

[...]

```

### Task 3.4: Test again for hivemaster

We are going to test that the warning is not showing anymore for our `hivemaster` host:

```
student@ansible-00-01-hivemaster:~$ ansible -i hosts ansible-00-01-hivemaster -m ping -u student --ask-pass
SSH password:
ansible-00-01-hivemaster | SUCCESS => {
  "ansible_facts": {
    "discovered_interpreter_python": "/usr/bin/python3"
  },
  "changed": false,
  "ping": "pong"
}
```

### Task 3.5: Check the connection user used by Ansible

To make sure that the user is “student”, we can run also the “id” command using `shell` module:

```
student@ansible-00-01-hivemaster:~$ ansible -i hosts all -m shell -a "id" -u student --ask-pass
SSH password:
ansible-00-01-hivemaster | CHANGED | rc=0 >>
uid=1002(student) gid=1003(student) groups=1003(student)

ansible-00-02-ubuntu | CHANGED | rc=0 >>
uid=1002(student) gid=1003(student) groups=1003(student)

10.142.15.213 | CHANGED | rc=0 >>
uid=1001(student) gid=1002(student) groups=1002(student)
context=unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023
```

Notice that we used here another module called `shell` to run the `id` command, instead of `raw` module which we used in [Task 1](#) (we are going to learn more about differences between these modules into modules lab).

### Task 3.6: Accessing system files (privilege escalation)

Firstly, we are going to try accessing `/etc/shadow` file using `student` user:

```
student@ansible-00-01-hivemaster:~$ ansible -i hosts all -m shell -a "cat /etc/shadow" -u student --ask-pass
SSH password:
ansible-00-02-ubuntu | FAILED | rc=1 >>
cat: /etc/shadow: Permission deniednon-zero return code

ansible-00-01-hivemaster | FAILED | rc=1 >>
cat: /etc/shadow: Permission deniednon-zero return code

10.142.15.213 | FAILED | rc=1 >>
cat: /etc/shadow: Permission deniednon-zero return code
```

As you may already expected, student user cannot access `/etc/shadow` file, so we have to instruct Ansible to become `root` before executing this command:

```
student@ansible-00-01-hivemaster:~$ ansible -i hosts all -m shell -a
"cat /etc/shadow" -u student --ask-pass --become
SSH password:
ansible-00-01-hivemaster | FAILED | rc=-1 >>
Missing sudo password
ansible-00-02-ubuntu | FAILED | rc=-1 >>
Missing sudo password
10.10.17.146 | FAILED | rc=-1 >>
Missing sudo password
```

It looks like we didn't provided the BECOME password for becoming `root` user. This can be fixed by adding `--ask-become-pass`.

Notice that the `BECOME` password defaults to `SSH` password, so in our case (because we use the same password) we can just press ENTER:

```
student@ansible-00-01-hivemaster:~$ ansible -i hosts all -m shell -a
"cat /etc/shadow" -u student --ask-pass --become --ask-become-pass
SSH password:
BECOME password[defaults to SSH password]:
ansible-00-01-hivemaster | CHANGED | rc=0 >>
root:$6$27488$DYVSMYCFtpga/pAdoxJM9iyaVI4BE0ku8zb2c7GWrPE28t5K0qwvaMvKez
jWUCIGdwDtWbrHnPn6kYse7SEjb1:18223:0:99999:7:::
daemon*:18213:0:99999:7:::
bin*:18213:0:99999:7:::
[...]

ansible-00-02-ubuntu | CHANGED | rc=0 >>
root:$6$45423$/USc3nLyxSEx0HmuXUCCi6g0TsqtLk9eDUkmvEQHuw1vHXLrpCZa5e3TfK
9De7c2H22Ie/3qvTCPKTX6WTO301:18223:0:99999:7:::
daemon*:18213:0:99999:7:::
[...]

10.142.15.213 | CHANGED | rc=0 >>
root:$6$48153$AYWTuD52gImgis/lWOyeY0LsTaK9byanOBfkCL97SImtVUYHjVzdying0J
KNgE34bgesc.D2fjtkinDSCrpjI0:18223:0:99999:7:::
bin*:17834:0:99999:7:::
[...]
```

#### Task 4: Running Ad-Hoc commands using key-based authentication

In everyday use the best practice is to have a public key installed on the remote hosts and it is also recommended to create a dedicated user who should have access using that key. Right now, we are going to create a new keypair for our student user and add the public key to the proper file `/home/student/.ssh/authorized_keys`. Notice that you can issue this command on any server you want, the key is provisioned to all 3 servers.



```
student@ansible-00-02-ubuntu:~$ cat /home/student/.ssh/authorized_keys
ssh-rsa
AAAAB3NzaC1yc2EAAAABIwAAAQEAKdlJG4MF1YUglv6cFYDE93gAK7Z3/HeFJ/ZjWgHIYQAt
NCkhHnR0EXlN3kJJUt/xP7zLwcfIbY73pjTVMtmwx+kheb1R4Uqj1SPTHQWrVpeddVDLmflS
h5nCFVJJZ9bdi8YTgyl2NKTbNIQ7U9fXP/AIaNHZBT0rBUcf055qCm+ub2qx+TOvkzalaIPz
+XHo6Cx37+Pkq3WmQBfDmAuOwRByE95bF1FYd02mGGvOiedK1bJM8jxcMTzqZ/0gyFA4cXkh
pQBjbMIEeWqpbE0ZUZ7RY+thHW2JIxfu0tcqjbUnMIkAwN+JDVBolRPuNKJ/4vpIMf3sfysg
n7dgrvS0tw== bogd@cloudbrain
```

Notice that this file already contains a public key, which was used during provisioning of the infrastructure, so be careful just to append the new key to this file.

#### Task 4.1: Generate a SSH keypair

In order to test key-based authentication we are going to generate a new public/private key pair, using RSA type (-t), 2048 length and save it as `ansible_key` in the current directory (without password):

```
student@ansible-00-01-hivemaster:~$ ssh-keygen -t rsa -b 2048 -f
ansible_key
Generating public/private rsa key pair.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in ansible_key.
Your public key has been saved in ansible_key.pub.
The key fingerprint is:
SHA256:gm+P8xHI5tl55qCJF88jmDj/9C4Z25rzLTzQSrZmO/4 student@ansible-00-
01-hivemaster
The key's randomart image is:
+---[RSA 2048]-----+
|
|
|
|   o .
|  . =.S
|   +*+.o
|  . =*&= o
| o o+#BOB
| oo*X&E+o
+---[SHA256]-----+
```

You will notice that 2 new files were created in the current directory (`/home/student`):

```
student@ansible-00-01-hivemaster:~$ ls | grep ansible
ansible_key
ansible_key.pub
```

Right now we are going to add the new generated public key (`ansible_key.pub`) to our student user on hivemaster host:

```
student@ansible-00-01-hivemaster:~$ cat ansible_key.pub >>
/home/student/.ssh/authorized_keys
```

To make things easier, we are going to use Ansible itself to add the public key also on the other 2 managed hosts (ubuntu and centos):

#### Task 4.2: Put publickey into correct place

As we just generated the new keypair, we need to put it in the corresponding file for student user (/home/student/.ssh/authorized\_keys).

Firstly, we have to copy the file also to the remote servers (we are going to use Ansible for that):

```
student@ansible-00-01-hivemaster:~$ ansible -i hosts all -m copy -a
"src=/home/student/ansible_key.pub dest=/home/student/" -u student --
ask-pass --become --ask-become-pass
```

Now that we have `ansible_key.pub` in `/home/student` let's append the key to the `authorized_keys` file from each host:

```
student@ansible-00-01-hivemaster:~$ ansible -i hosts all -m shell -a
"cat ansible_key.pub >> /home/student/.ssh/authorized_keys" -u student -
-ask-pass --become --ask-become-pass
SSH password:
BECOME password[defaults to SSH password]:
ansible-00-02-ubuntu | CHANGED | rc=0 >>

ansible-00-01-hivemaster | CHANGED | rc=0 >>

10.142.15.213 | CHANGED | rc=0 >>
```

You can check that the public key was properly added:

```
student@ansible-00-01-hivemaster:~$ cat
/home/student/.ssh/authorized_keys
ssh-rsa
AAAAB3NzaC1yc2EAAAABIwAAAQEAKdlJG4MF1YUglv6cFYDE93gAK7Z3/HeFJ/ZjWgHIYQAt
NCkhHnR0EXlN3kJJUt/xP7zLwcfIbY73pjTVMtmwx+kheb1R4Uqj1SPTHQWrVpeddVDLmfLS
h5nCFVJJZ9bdi8YTgyl2NKTBNiQ7U9fXP/AIaNHZBT0rBUcf055qCm+ub2qx+TOvkzalaIPz
+XHo6Cx37+Pkq3WmQBfDmAuOwRByE95bF1FYd02mGGvOiedK1bJM8jxcMTzqZ/0gyFA4cXkh
pQBjbMIEeWqpbE0ZUZ7RY+thHW2JIxfu0tcqjbUnMIkAwN+JDVBolRPuNKJ/4vpIMf3sfysg
n7dgrvS0tw== bogd@thermite
ssh-rsa
AAAAB3NzaC1yc2EAAAADAQABAAQD76mY0aD+iWPoR5zc+1Vp7Ve6vGSRbliogXl8ABA0l
urw8GKMnQHNBCAgNMO+u1R2qRRgY/fIBXF1D4iQm1Dd4Z4c3qwbbZAltK7zsoDcY8AZZwj5t
f54yhqN3BGQMySABIBWF7FKjyH1iZ/ut3hA2hqlIDv2rpqzB+U2Bov4+B7L3naWMAYYEnGtK
SD75GKsSchsNpjVfZpw5v6V/3tMBw1iPS55IjWDTNwSCKrOmaFsxU45oMN7HnZCh5K1zL3+z1
```

```
w5golr5Smz+JgtMAAzhUgb9gDz9dbaJnrFoU6bS/wKNBebjrk3LPTMHSnW62pi/YtxxpRw2S
Qa5QRQnizgsn student@ansible-00-01-hivemaster
```

#### Task 4.3: Ping all hosts (using key-based auth)

It's time to test the key-based authentication, specifying user (-u) and private key file (--private-key):

```
student@ansible-00-01-hivemaster:~$ ansible -i hosts all -m ping -u
student --private-key /home/student/ansible_key
ansible-00-02-ubuntu | SUCCESS => {
  "ansible_facts": {
    "discovered_interpreter_python": "/usr/bin/python3"
  },
  "changed": false,
  "ping": "pong"
}
ansible-00-01-hivemaster | SUCCESS => {
  "ansible_facts": {
    "discovered_interpreter_python": "/usr/bin/python3"
  },
  "changed": false,
  "ping": "pong"
}
10.142.15.213 | SUCCESS => {
  "ansible_facts": {
    "discovered_interpreter_python": "/usr/bin/python"
  },
  "changed": false,
  "ping": "pong"
}
```

#### Task 4.4: Accessing system files (privilege escalation)

Also using key-based authentication we have to instruct Ansible to become root (--become) and to prompt for become password (--ask-become-pass):

```
student@ansible-00-01-hivemaster:~$ ansible -i hosts all -m shell -a
"cat /etc/shadow" -u student --private-key /home/student/ansible_key --
become --ask-become-pass
ansible-00-02-ubuntu | CHANGED | rc=0 >>
root:$6$45423$/USc3nLyxSEx0HmuXUCCi6g0TsqTLk9eDUkmvEQHuwlVHXLrpCZa5e3TfK
9De7c2H22Ie/3qvTCPKTX6WTO301:18223:0:99999:7:::
daemon*:18213:0:99999:7:::
[...]

ansible-00-01-hivemaster | CHANGED | rc=0 >>
root:$6$27488$DYVSMYCFtpga/pAdoxJM9iyaVI4BE0ku8zb2c7GWrPE28t5K0qwvaMvKez
jWUCIGdwDtWbrHnPn6kYse7SEjb1:18223:0:99999:7:::
[...]
```

```
10.142.15.213 | CHANGED | rc=0 >>
root:$6$48153$AYWTuD52gImgis/lWOyeY0LsTaK9byanOBFkCL97SImtVUyHjVzdyinq0J
KNgE34bgesc.D2fjtkinDSCrpjI0:18223:0:99999:7:::
[...]
```

#### Task 4.5: Command module

We are going to explore this module right now, because command module is the default module for ansible command line utility. Let's test rebooting our ubuntu server (notice that we omitted specifying module name because Ansible defaults to command module):

```
student@ansible-00-01-hivemaster:~$ ansible -i hosts ansible-00-02-ubuntu -a "/sbin/reboot" -u student --private-key /home/student/ansible_key --become --ask-become-pass
BECOME password:
ansible-00-02-ubuntu | UNREACHABLE! => {
  "changed": false,
  "msg": "Failed to connect to the host via ssh: ssh: connect to host ansible-00-02-ubuntu port 22: Connection timed out",
  "unreachable": true
}
```

Soon, you will notice a "Connection timed out" error, as the server become UNREACHABLE. Let's wait for server to reboot and test again the ping:

```
student@ansible-00-01-hivemaster:~$ ansible -i hosts ansible-00-02-ubuntu -m ping -u student --private-key /home/student/ansible_key
ansible-00-02-ubuntu | SUCCESS => {
  "ansible_facts": {
    "discovered_interpreter_python": "/usr/bin/python3"
  },
  "changed": false,
  "ping": "pong"
}
```

#### Task 4: Setup module in Ad-Hoc commands

Facts represent discovered variables about hosts gathered by Ansible using setup module. When using ad-hoc commands, we have to explicitly instruct Ansible to gather facts:

```
student@ansible-00-01-hivemaster:~$ ansible -i hosts ansible-00-02-ubuntu -m setup -u student --private-key /home/student/ansible_key
ansible-00-02-ubuntu | SUCCESS => {
  "ansible_facts": {
    "ansible_all_ipv4_addresses": [
      "10.128.0.49"
    ],

```

[...]

As you may see in your environment, the amount of gathered facts (just for one host – notice that we run the command only against **ubuntu** host) is enormous, so it very likely that you may want to narrow the results to see only some of them. We can use `filter` option to make this possible:

```
student@ansible-00-01-hivemaster:~$ ansible -i hosts ansible-00-02-ubuntu -m setup -a "filter=ansible_env" -u student --private-key /home/student/ansible_key
ansible-00-02-ubuntu | SUCCESS => {
  "ansible_facts": {
    "ansible_env": {
      "HOME": "/home/student",
      "LANG": "C.UTF-8",
      "LOGNAME": "student",
      "MAIL": "/var/mail/student",
      "PATH":
"/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/usr/local/games",
      "PWD": "/home/student",
      "SHELL": "/bin/bash",
      "SHLVL": "1",
      "SSH_CLIENT": "10.128.0.48 52772 22",
      "SSH_CONNECTION": "10.128.0.48 52772 10.128.0.49 22",
      "SSH_TTY": "/dev/pts/0",
      "TERM": "xterm",
      "USER": "student",
      "XDG_RUNTIME_DIR": "/run/user/1002",
      "XDG_SESSION_ID": "6",
      "_": "/bin/sh"
    },
    "discovered_interpreter_python": "/usr/bin/python3"
  },
  "changed": false
}
```