



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

Preparing hosts for Ansible use

## Table of Contents

<b>Lab Overview and objectives</b>	<b>2</b>
<i>Guided Tasks</i>	2
Task 1: Create a dedicated user	2
Task 2: Configuring key-based auth for “ansible” user	2
Task 2.1: Create .ssh folder in /home/ansible	2
Task 2.2: Copy public key to /home/ansible/.ssh/authorized_keys	4
Task 2.3: Test key-based auth for ‘ansible’ user	5
Task 3: Giving the ‘ansible’ user sudo permissions	6
Task 4: Configuring remote_user and private_key in ansible.cfg	6
Task 5: Configuring static host inventories	8
Task 5.1: Explore inventory file	8
Task 5.2: Create a new inventory file (INI format)	8
Task 5.3: Test the inventory file	9
Task 5.4: Create a new inventory file (YAML format)	11
Task 6: Set the default inventory	12
Task 7: Test “ansible” command	12

# Lab Overview and objectives

In this lab we are going to perform a full setup of our hosts in order to be prepared for managing with Ansible playbooks. We are going to create a dedicated user, configure key-based authentication and also provide **sudo** permissions for this user. Next we will perform some changes in ansible configuration file and we will also create a static host inventory file.

## Guided Tasks

### Task 1: Create a dedicated user

As we mentioned in previous labs, it is highly recommended to have a dedicated user for managing hosts with Ansible, so we are going to create a new user called “ansible”. To perform this task we can use `useradd` command with `-m` to create the user’s home directory and `-s /bin/bash` to set the default shell. You can add the users manually, or you can use Ansible to automate this for all hosts (login with student user using password authentication or key-based authentication):

```
student@ansible-00-01-hivemaster:~$ ansible -i hosts all -m user -a  
"name=ansible state=present" --ask-pass --become --ask-become-pass
```

Otherwise, you can login into every host using SSH, and run the command which was provided as argument for `shell` module:

```
sudo useradd -m ansible -s /bin/bash
```

### Task 2: Configuring key-based auth for “ansible” user

Now that we have just created our user, we have to setup SSH key-based authentication. We are going to use the same keypair generated in the previous lab, which is located in `/home/student`.

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

Things are similar to the previous task, we can make this manually, or we can use Ansible.

#### Task 2.1: Create `.ssh` folder in `/home/ansible`

As the name of the task says, we are going to create `/home/ansible/.ssh` folder with owner/group `ansible:ansible` and `0700` permissions (`rxw` for owner).

In this subtask we will use Ansible itself to perform the changes, but you can also use the following commands for each host (if you don't feel comfortable with Ansible modules yet):

```
sudo mkdir /home/ansible/.ssh
sudo chown ansible:ansible /home/ansible/.ssh
sudo chmod 700 /home/ansible/.ssh
```

The corresponding Ansible command is:

```
student@ansible-00-01-hivemaster:~$ ansible -i hosts all -m file -a
"path=/home/ansible/.ssh state=directory owner=ansible group=ansible
mode=0700" --ask-pass --become --ask-become-pass
SSH password:
BECOME password[defaults to SSH password]:
ansible-00-02-ubuntu | CHANGED => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python3"
    },
    "changed": true,
    "gid": 1005,
    "group": "ansible",
    "mode": "0700",
    "owner": "ansible",
    "path": "/home/ansible/.ssh",
    "size": 4096,
    "state": "directory",
    "uid": 1004
}
ansible-00-01-hivemaster | CHANGED => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python3"
    },
    "changed": true,
    "gid": 1005,
    "group": "ansible",
    "mode": "0700",
    "owner": "ansible",
    "path": "/home/ansible/.ssh",
    "size": 4096,
    "state": "directory",
    "uid": 1004
}
10.142.15.213 | CHANGED => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python"
    },
    "changed": true,
    "gid": 1004,
    "group": "ansible",
    "mode": "0700",
```

```
"owner": "ansible",
"path": "/home/ansible/.ssh",
"secontext": "unconfined_u:object_r:ssh_home_t:s0",
"size": 6,
"state": "directory",
"uid": 1003
}
```

## Task 2.2: Copy public key to /home/ansible/.ssh/authorized\_keys

Remember that we have already copied the public key (ansible\_key.pub) to each server on /home/student. So, we just have to copy it to `authorized_keys` file and set proper owner/group and permissions:

```
student@ansible-00-01-hivemaster:~$ sudo -i
root@ansible-00-01-hivemaster:~# cat /home/student/ansible_key.pub >>
/home/ansible/.ssh/authorized_keys
root@ansible-00-01-hivemaster:~# chown ansible:ansible
/home/ansible/.ssh/authorized_keys
root@ansible-00-01-hivemaster:~# chmod 644
/home/ansible/.ssh/authorized_keys
root@ansible-00-01-hivemaster:~# exit
logout
student@ansible-00-01-hivemaster:~$
```

Notice that we jumped to `root` account to be able to write inside `.ssh` folder of “ansible” user!

Otherwise, this can be also done using Ansible using `copy` module (notice that we used an interesting parameter for this module, called `remote_src`, which specifies if the source file is copied from remote or from the local host to destination):

```
student@ansible-00-01-hivemaster:~$ ansible -i hosts all -m copy -a
"src=/home/student/ansible_key.pub remote_src=yes
dest=/home/ansible/.ssh/authorized_keys owner=ansible group=ansible
mode=0644" --ask-pass --become --ask-become-pass
SSH password:
BECOME password[defaults to SSH password]:
ansible-00-02-ubuntu | CHANGED => {
  "ansible_facts": {
    "discovered_interpreter_python": "/usr/bin/python3"
  },
  "changed": true,
  "checksum": "92e5c547a77ee942116b4b7295d35afb57e8b9db",
  "dest": "/home/ansible/.ssh/authorized_keys",
  "gid": 1005,
  "group": "ansible",
  "md5sum": "d869316f5c31572b8232b1bee60e8313",
  "mode": "0644",
  "owner": "ansible",
  "size": 414,
```

```

    "src": "/home/student/ansible_key.pub",
    "state": "file",
    "uid": 1004
  }
ansible-00-01-hivemaster | CHANGED => {
  "ansible_facts": {
    "discovered_interpreter_python": "/usr/bin/python3"
  },
  "changed": true,
  "checksum": "92e5c547a77ee942116b4b7295d35afb57e8b9db",
  "dest": "/home/ansible/.ssh/authorized_keys",
  "gid": 1005,
  "group": "ansible",
  "md5sum": "d869316f5c31572b8232b1bee60e8313",
  "mode": "0644",
  "owner": "ansible",
  "size": 414,
  "src": "/home/student/ansible_key.pub",
  "state": "file",
  "uid": 1004
}
10.142.15.213 | CHANGED => {
  "ansible_facts": {
    "discovered_interpreter_python": "/usr/bin/python"
  },
  "changed": true,
  "checksum": "92e5c547a77ee942116b4b7295d35afb57e8b9db",
  "dest": "/home/ansible/.ssh/authorized_keys",
  "gid": 1004,
  "group": "ansible",
  "md5sum": "d869316f5c31572b8232b1bee60e8313",
  "mode": "0644",
  "owner": "ansible",
  "secontext": "unconfined_u:object_r:ssh_home_t:s0",
  "size": 414,
  "src": "/home/student/ansible_key.pub",
  "state": "file",
  "uid": 1003
}

```

### Task 2.3: Test key-based auth for 'ansible' user

```

student@ansible-00-01-hivemaster:~$ ansible -i hosts all -m ping -u
ansible --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"
}
```

In order to make sure that the correct user was used, we can also run “id” command:

```
student@ansible-00-01-hivemaster:~$ ansible -i hosts all -a "id" -u
ansible --private-key /home/student/ansible_key
ansible-00-02-ubuntu | CHANGED | rc=0 >>
uid=1004(ansible) gid=1005(ansible) groups=1005(ansible)

ansible-00-01-hivemaster | CHANGED | rc=0 >>
uid=1004(ansible) gid=1005(ansible) groups=1005(ansible)

10.142.15.213 | CHANGED | rc=0 >>
uid=1003(ansible) gid=1004(ansible) groups=1004(ansible)
context=unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023
```

### Task 3: Giving the ‘ansible’ user sudo permissions

The ‘ansible’ user should be able to run commands as root, without entering the password, so we have to add it to `/etc/sudoers` file. The recommended way to perform changes on sudoers file is using `visudo` command, as it performs also a check before saving, in order not to mess something and not be able to edit this file anymore:

```
student@ansible-00-01-hivemaster:~$ sudo visudo
#add this line at the end of the file
ansible ALL=(ALL:ALL) NOPASSWD: ALL
```

Make sure to login also in the other 2 servers and perform the same task.

### Task 4: Configuring `remote_user` and `private_key` in `ansible.cfg`

In [Task 2.3](#) we tested key-based authentication for our new ‘ansible’ user, specifying its name and path to the private key. Notice that it is also necessary in case of using playbooks, not just in case of ad-hoc

commands. We can configure default values for these parameters in `/etc/ansible/ansible.cfg` in order to avoid writing them every time we run ansible:

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

#Uncomment and modify (or add) the following lines:
remote_user = ansible
private_key_file = /home/student/ansible_key
```

Right now we can check that these parameters are set correctly:

```
student@ansible-00-01-hivemaster:~$ ansible -i hosts all -m shell -a
"whoami"
ansible-00-02-ubuntu | CHANGED | rc=0 >>
ansible

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

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

In the same file, `ansible.cfg`, we can also set (enable) some other parameters like `become`, `become_user`, `become_method` in `privilege_escalation` section:

```
[privilege_escalation]
#become=True
#become_method=sudo
#become_user=root
#become_ask_pass=False
```

Notice that if we set `become=True` all the tasks will be executed as `become_user` (root by default):

```
student@ansible-00-01-hivemaster:~$ ansible -i hosts all -m shell -a
"whoami"
ansible-00-02-ubuntu | CHANGED | rc=0 >>
root

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

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

Make sure to comment again `become=True` because right now we don't want that all the requests to be executed as root user.



## Task 5: Configuring static host inventories

Until now we worked with a basic inventory file which contains only the hostnames (IP for centos) of our hosts. In this section we are going to make this inventory a little bit more complex (not by adding more hosts) by grouping hosts in groups and setting some variables for them.

### Task 5.1: Explore inventory file

Before performing any changes to our inventory file let's see how Ansible sees our `hosts` file. We can use `ansible-inventory` command using `--graph` or `--list`:

```
student@ansible-00-01-hivemaster:~$ ansible-inventory -i hosts --list
{
  "_meta": {
    "hostvars": {}
  },
  "all": {
    "children": [
      "ungrouped"
    ]
  },
  "ungrouped": {
    "hosts": [
      "10.142.15.213",
      "ansible-00-01-hivemaster",
      "ansible-00-02-ubuntu"
    ]
  }
}
```

```
student@ansible-00-01-hivemaster:~$ ansible-inventory -i hosts --graph
@all:
|--@ungrouped:
|   |--10.142.15.213
|   |--ansible-00-01-hivemaster
|   |--ansible-00-02-ubuntu
```

As you can see from these results, our servers are listed in “ungrouped” group.

### Task 5.2: Create a new inventory file (INI format)

Our first basic inventory file was written in INI format, so we are going to start also with this format for the current inventory. Let's put our hosts in groups according to the following table:

Group	Hosts	Group Vars
webservers	hivemaster, centos	apache_port 80, apache_path /var/www/html/
dbservers	hivemaster, ubuntu	

We can also group these 2 groups into a bigger group (let's name it datacenter).

```
student@ansible-00-01-hivemaster:~$ vi hosts_ini
hivemaster ansible_host="10.128.0.48" ansible_user=ansible
ansible_ssh_private_key_file=/home/student/ansible_key
ubuntu ansible_host="10.128.0.49"
centos ansible_host="10.142.15.213" ansible_user=student

[webservers]
ubuntu
centos

[webservers:vars]
apache_port=80
apache_path=/var/www/html/

[dbservers]
ubuntu
hivemaster

[datacenter:children]
webservers
dbservers
```

Notice some important aspects in this inventory file:

- in the first part we defined the `hosts` (using `inventory_hostnames`) and some `host_vars` (`ansible_host` – this can also be the hostname instead of IP address, `ansible_user`, `ansible_ssh_private_key_file`); these are redundant for the first 2 hosts as they are already defined in `ansible.cfg`;
- we created the specified groups (`webservers`, `dbservers`), included the hosts into them and set the `group_vars`;
- we included both groups into the bigger one, `datacenter`, using “`:children`”, otherwise Ansible would try to include in `datacenter` group 2 hosts called `webservers` and `dbservers`.

### Task 5.3: Test the inventory file

It's time to test our new inventory, so let's filter gathered facts to see some info about IP addresses:

```
student@ansible-00-01-hivemaster:~$ ansible -i hosts_ini all -m setup -a
"filter=*ipv4"
ubuntu | SUCCESS => {
  "ansible_facts": {
    "ansible_default_ipv4": {
      "address": "10.128.0.49",
      "alias": "ens4",
      "broadcast": "global",
      "gateway": "10.128.0.1",
```

```

        "interface": "ens4",
        "macaddress": "42:01:0a:80:00:31",
        "mtu": 1460,
        "netmask": "255.255.255.255",
        "network": "10.128.0.49",
        "type": "ether"
    },
    "discovered_interpreter_python": "/usr/bin/python3"
},
"changed": false
}
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
}
centos | SUCCESS => {
    "ansible_facts": {
        "ansible_default_ipv4": {
            "address": "10.142.15.213",
            "alias": "eth0",
            "broadcast": "10.142.15.213",
            "gateway": "10.142.0.1",
            "interface": "eth0",
            "macaddress": "42:01:0a:8e:0f:d5",
            "mtu": 1460,
            "netmask": "255.255.255.255",
            "network": "10.142.15.213",
            "type": "ether"
        },
        "discovered_interpreter_python": "/usr/bin/python"
    },
    "changed": false
}

```

Right now we want also to see what user is using Ansible to connect to every host (remember that for centos we set the “student” user, for hivemaster we set also the `ansible_user` and `ansible_ssh_key`, while for ubuntu we didn’t set any host var):

```
student@ansible-00-01-hivemaster:~$ ansible -i hosts_ini all -a "whoami"
ubuntu | CHANGED | rc=0 >>
ansible

hivemaster | CHANGED | rc=0 >>
ansible

centos | CHANGED | rc=0 >>
student
```

As you can see, for `hivemaster` and `ubuntu` the “**ansible**” user was used (this is also configured in `ansible.cfg`), while for `centos` the “**student**” user was used, so the variable defined in the inventory file has precedence over the `ansible_user` defined in `ansible.cfg`!

Now we want to set also for centos the ansible user (so we have to remove the `ansible_user=student` for centos from inventory file, or replace it with ‘ansible’).

```
student@ansible-00-01-hivemaster:~$ vi hosts_ini
centos ansible_host="10.142.15.213" ansible_user=ansible
```

Test again using `ansible -i hosts_ini all -a "whoami"`.

#### Task 5.4: Create a new inventory file (YAML format)

Right now you just have to create the same inventory file, using YAML format (basically it’s just a rewrite of the same inventory):

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

webservers:
  hosts:
    ubuntu:
      ansible_host: "10.128.0.49"
    centos:
      ansible_host: "10.142.15.213"
      ansible_user: ansible #we replaced student with ansible
  vars:
    apache_port: 80
    apache_path: /var/www/html/

dbservers:
  hosts:
    ubuntu:
    hivemaster:
      ansible_host: "10.128.0.48"
      ansible_user: ansible
```

```
ansible_ssh_private_key_file: /home/student/ansible_key

datacenter:
  children:
    webservers:
    dbservers:
```

#### Task 5.5: Explore the new inventory files (INI and YAML)

We can use the same command “ansible-inventory” to explore the list and graph of the new inventory files:

```
student@ansible-00-01-hivemaster:~$ ansible-inventory -i hosts_ini --
list
student@ansible-00-01-hivemaster:~$ ansible-inventory -i hosts_ini --
graph
student@ansible-00-01-hivemaster:~$ ansible-inventory -i hosts_yaml --
list
student@ansible-00-01-hivemaster:~$ ansible-inventory -i hosts_yaml --
graph
```

#### Task 6: Set the default inventory

Ansible uses by default the inventory file located at /etc/ansible/hosts. This file is commented by default and it is also a good start to explore its content in order to make an idea about the various possibilities of writing the inventory file:

```
student@ansible-00-01-hivemaster:~$ cat /etc/ansible/hosts
# This is the default ansible 'hosts' file.
#
# It should live in /etc/ansible/hosts
[...]
```

Let's copy the content of our hosts\_ini to /etc/ansible/hosts:

```
student@ansible-00-01-hivemaster:~$ sudo bash -c 'cat hosts_ini >
/etc/ansible/hosts'

student@ansible-00-01-hivemaster:~$ cat /etc/ansible/hosts
hivemaster ansible_host="10.128.0.48" ansible_user=ansible
ansible_ssh_private_key_file=/home/student/ansible_key
ubuntu ansible_host="10.128.0.49"
centos ansible_host="10.142.15.213" ansible_user=ansible
[...]
```

#### Task 7: Test “ansible” command

Finally, we can test using `ansible` command without providing host/user/key options:

```
student@ansible-00-01-hivemaster:~$ ansible all -m ping
ubuntu | SUCCESS => {
  "ansible_facts": {
    "discovered_interpreter_python": "/usr/bin/python3"
  },
  "changed": false,
  "ping": "pong"
}
hivemaster | SUCCESS => {
  "ansible_facts": {
    "discovered_interpreter_python": "/usr/bin/python3"
  },
  "changed": false,
  "ping": "pong"
}
centos | SUCCESS => {
  "ansible_facts": {
    "discovered_interpreter_python": "/usr/bin/python"
  },
  "changed": false,
  "ping": "pong"
}
```

```
student@ansible-00-01-hivemaster:~$ ansible all -a "id"
ubuntu | CHANGED | rc=0 >>
uid=1004(ansible) gid=1005(ansible) groups=1005(ansible)

hivemaster | CHANGED | rc=0 >>
uid=1004(ansible) gid=1005(ansible) groups=1005(ansible)

centos | CHANGED | rc=0 >>
uid=1003(ansible) gid=1004(ansible) groups=1004(ansible)
context=unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023
```