ĐẠI HỌC QUỐC GIA TP.HCM

TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN

MÔN HỌC: QUẢN TRỊ MẠNG VÀ HỆ THỐNG

BÁO CÁO ĐỒ ÁN

System Automation

Giảng viên hướng dẫn: Trần Thị Dung

NT132.O11.ATCL

Sinh viên thực hiện :

21522492 - Ngô Minh Quân

21522312 - Phùng Đức Lương

21522483 - Chu Nguyễn Hoàng Phương

TP.HCM,Ngày 4 tháng 1 năm 2024

Table of Contents

I. Introduction

1.1. General Information

- Ansible is a software tool that provides simple but powerful automation for cross-platform computer support. It is primarily intended for IT professionals, who use it for application deployment, updates on workstations and servers, cloud provisioning, configuration management, intra-service orchestration, and nearly anything a systems administrator does on a weekly or daily basis. Ansible doesn't depend on agent software and has no additional security infrastructure, so it's easy to deploy.

- Because Ansible is all about automation, it requires instructions to accomplish each job. With everything written down in simple script form, it's easy to do version control. While Ansible may be at the forefront of automation, systems administration, and DevOps, it's also useful to everyday users. Ansible allows us to configure not just one computer, but potentially a whole network of computers at once, and using it requires no programming skills. Instructions written for Ansible are human-readable. Whether we're entirely new to computers or an expert, Ansible files are easy to understand.

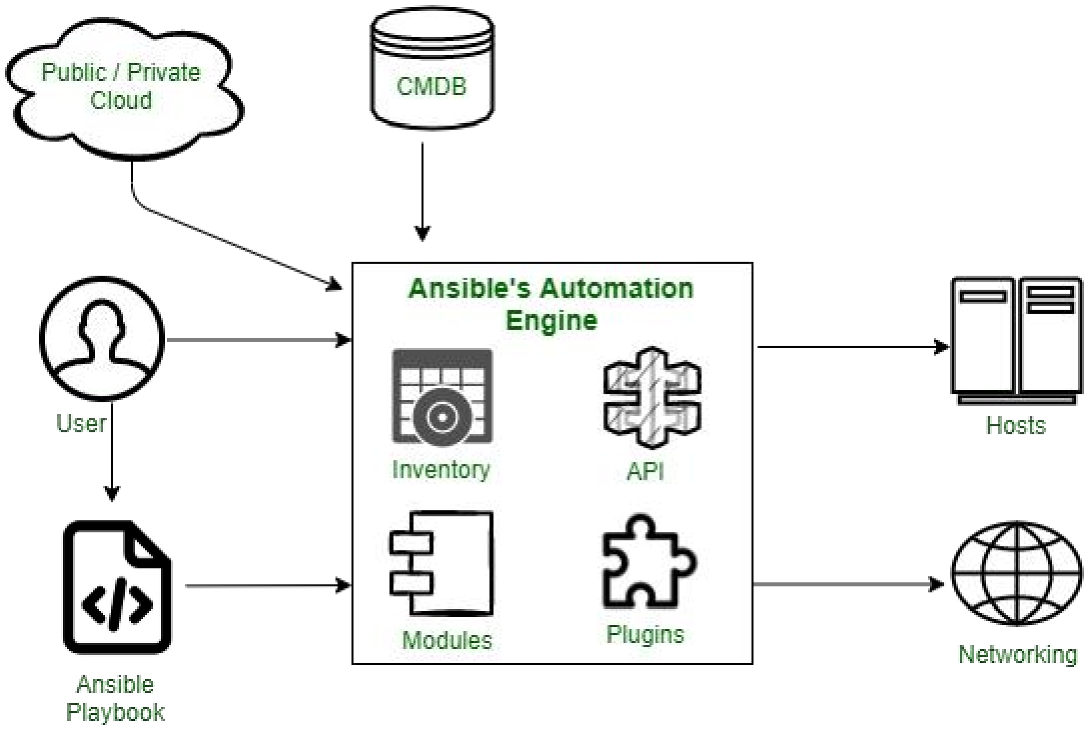
1.2. Component

- Ansible is divided into:

+ Automation Engine(Core) : Open Source.

+ Ansible Tower: Enterprise framework but also have UI, Restful API.

- The difference is that in the core part we have to use command-line, create scripts, while the Enterpise part has a more convenient UI for operation, etc. Here, we will discuss the architecture part of Ansible Core and will discuss its components. The Ansible automation engine consists of various components as described below as follows.



- Inventories
+ Ansible inventories are lists of hosts with their IP addresses, servers, and databases which have to be managed via an SSH for UNIX, Linux, or Networking devices, and WinRM for Windows systems.

- APIs
+ One may use application programming interfaces or APIs to enhance Ansible's connection choices. This covers more than just using SSH for transmission and extends to callbacks and other functionalities. The Ansible APIs serve as a conduit for public and private cloud applications.

- Modules
+ Modules are essential software that Ansible delivers from the command computer to all nodal network points or distant hosts. They are predetermined instructions that are executed directly on remote hosts. Playbooks run modules that manage applications, packages, and files. Ansible executes all modules for delivering updates or performing the required activity and then eliminates them after they're through. Ansible has over 450 modules for typical tasks. Ansible has hundreds of built-in modules, the pieces of code that are run when a playbook is launched. A playbook has plays, containing various tasks that include modules.

- Plugins
+ Plugins are little pieces of code that augment a website's functionality. Ansible comes with several of these, but one can create their own. Plugins are a specific type of module in this case. Before a module is performed on the nodes, the plugins are run. For logging reasons, plugins are executed on the primary control unit. We have call-back plugins because they allow us to connect to various Ansible events for display and reporting. To minimize the costs of fact-gathering

processes, cache plugins are used. Action plugins are front-end modules that perform operations on the controller system prior to invoking the modules directly.

- Networking
+ Ansible is used to automate different networks, and it uses the simple, secure, and powerful agentless automation framework for IT operations and development. It uses a type of data model which separated from the Ansible automation engine that spans the different hardware quite easily.

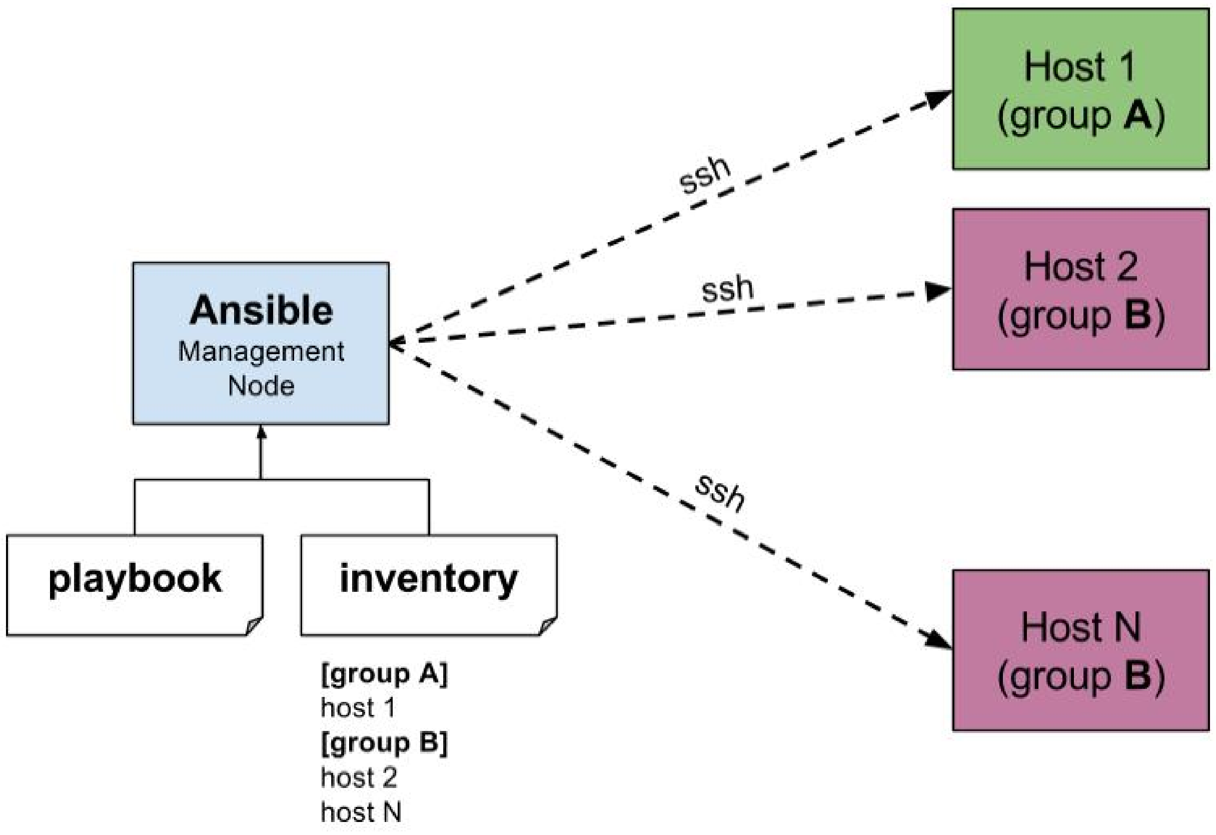
- Hosts
+ In the Ansible architecture, hosts are the node systems, which are automated by Ansible, and any machine such as RedHat, Linux, Windows, etc.

- Playbooks
+ Playbooks for Ansible are task-specific user guides. Playbooks dictate our workflow since functions written in them are executed in the order they are written. They are simple text documents created in YAML, a data serialization language that humans understand. They are at the heart of what makes Ansible so attractive since they describe the tasks one must perform quickly without requiring the user to remember particular terminology. In addition to being able to describe settings, they may also orchestrate the stages of any manually arranged task and conduct tasks concurrently or sequentially

- CMDB
+ It stands for Configuration Management Database (CMDB). In this, it holds data to a collection of IT assets, and it is a repository or data warehouse where we will store this kind of data, and It also defines the relationships between such assets. By deploying the Ansible-CMDB code, users may automatically transform the results of Ansible's data-collecting function into a static HTML summary page.

- Cloud
+ A cloud is a network of remote servers on which we can store, manage, and process the data. These servers are hosted on the internet and storing the data remotely rather than the local server. It just launches the resources and instances on the cloud, connect them to the servers, and we have good knowledge of operating our tasks remotely.

1.3. Operation



- In Ansible, there are two categories of computers: the control node and managed nodes. The control node is a computer that runs Ansible. There must be at least one control node, although a backup control node may also exist. A managed node is any device being managed by the control node.

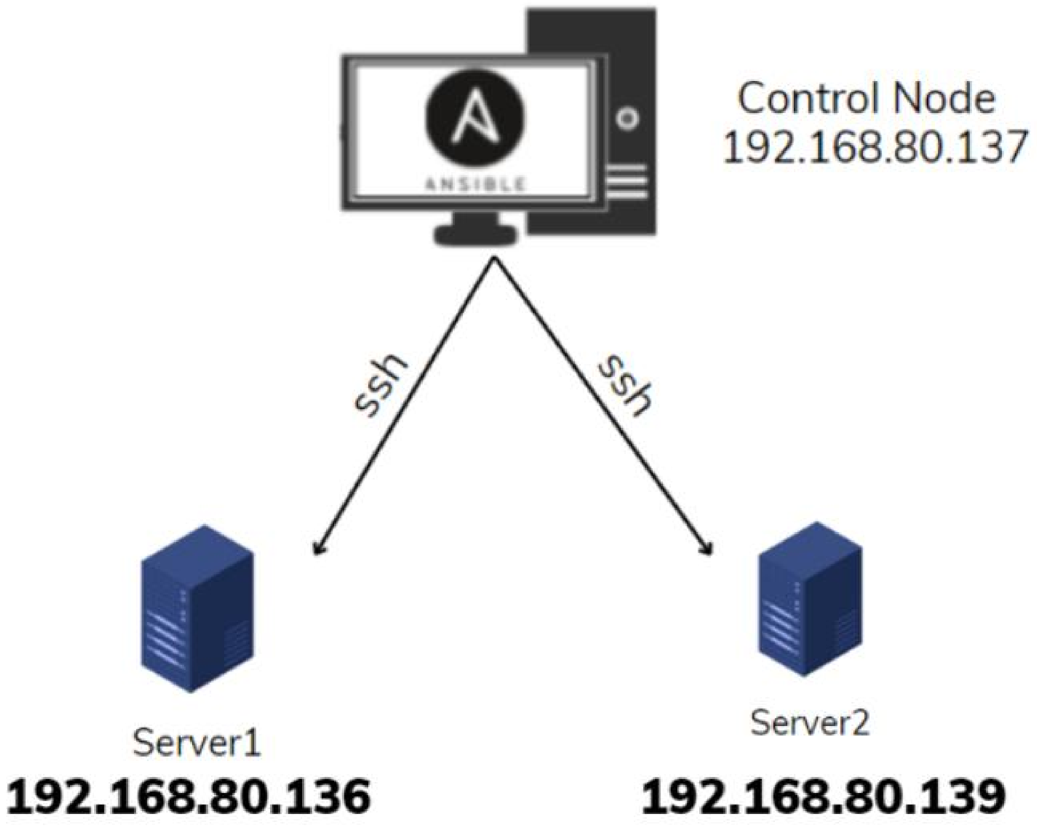
- Ansible interacts with our networks and sends little programs, known as modules, to them. These modules are utilized to complete automated tasks as systems designed to be resource models for the functioning at the desired state. Ansible runs these modules and eliminates them after they're done. If modules weren't available, we would have to depend on ad-hoc procedures and scripting to complete tasks. Ansible's management node is the primary node overseeing the Playbook's implementation.

- The management node sets up an SSH connection before executing the modules and installing the product on the host workstations. Once the modules have been deployed, it eliminates them. So that's how it works with Ansible. Python is used to create an Ansible script and connects remote hosts through SSH, specified in the inventory file. Ansible is agentless – implying that it doesn't need any program to be installed on the nodes it controls.

- Ansible takes inventory data to determine which machines we wish to control. It has a default inventory file, but users can customize it to manage the servers they want to. To link to servers and conduct tasks, Ansible employs the SSH protocol. Ansible establishes a connection with the remote system and distributes the modules required for command or playbook execution. Ansible uses human-readable YAML templates to allow the automation of repetitive processes without the need to master a complex programming language.

II. Implementation

2.1. Topology

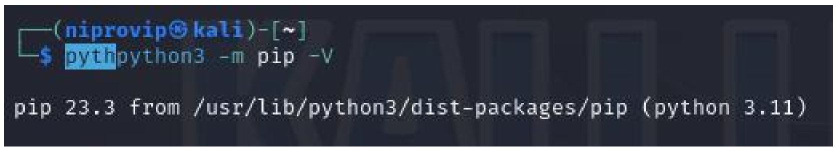


Name	IP	Software/Service
Control Node	192.168.80.137	Ansible
Server 1 (Managed Node)	192.168.80.136	Web Service (dvwa)
Server 2 (Managed Node)	192.168.80.139	Web Service (dvwa)

2.2. Installation

A. Installing and upgrading Ansible with pip

- To verify whether pip is already installed for our preferred Python: ***python3 -m pip -V***



```
┌──(niprovip㉿kali)-[~]
└─$ pythpython3 -m pip -V

pip 23.3 from /usr/lib/python3/dist-packages/pip (python 3.11)
```

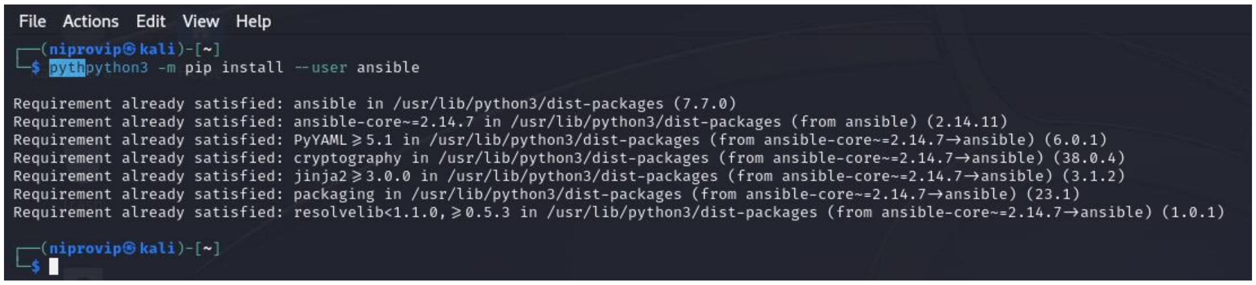
- If we see an error like No module named pip, we will need to install pip under our chosen Python interpreter before proceeding. Type these commands:

curl https://bootstrap.pypa.io/get-pip.py -o get-pip.py

python3 get-pip.py --user

- Next step, Use pip in our selected Python environment to install the full Ansible package for the current user:

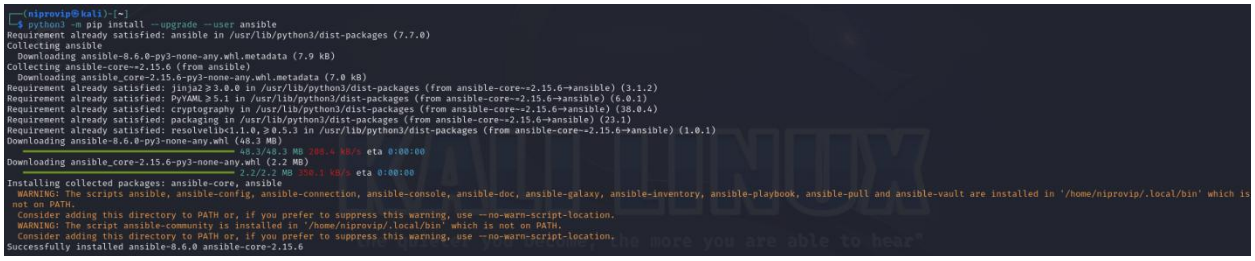
python3 -m pip install --user ansible



```
┌──(niprovip㉿kali)-[~]
└─$ pythpython3 -m pip install --user ansible

Requirement already satisfied: ansible in /usr/lib/python3/dist-packages (7.7.0)
Requirement already satisfied: ansible-core~=2.14.7 in /usr/lib/python3/dist-packages (from ansible) (2.14.11)
Requirement already satisfied: PyYAML≥5.1 in /usr/lib/python3/dist-packages (from ansible-core~=2.14.7→ansible) (6.0.1)
Requirement already satisfied: cryptography in /usr/lib/python3/dist-packages (from ansible-core~=2.14.7→ansible) (38.0.4)
Requirement already satisfied: jinja2≥3.0.0 in /usr/lib/python3/dist-packages (from ansible-core~=2.14.7→ansible) (3.1.2)
Requirement already satisfied: packaging in /usr/lib/python3/dist-packages (from ansible-core~=2.14.7→ansible) (23.1)
Requirement already satisfied: resolvelib<1.1.0,≥0.5.3 in /usr/lib/python3/dist-packages (from ansible-core~=2.14.7→ansible) (1.0.1)

┌──(niprovip㉿kali)-[~]
└─$ 
```

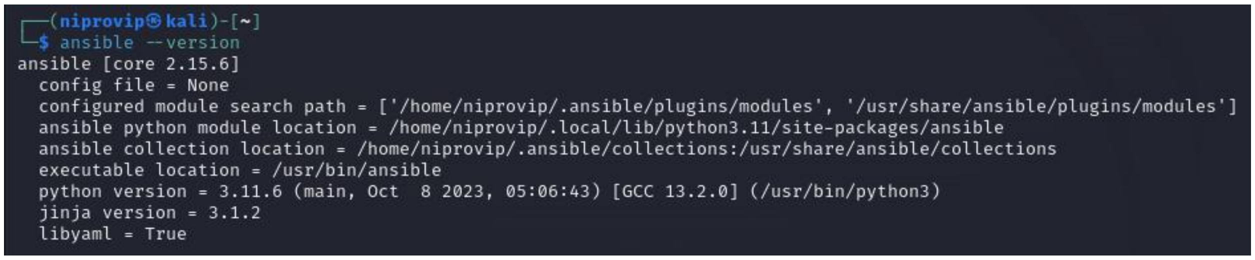
- To upgrade an existing Ansible installation in this Python environment to the latest released version, simply add --upgrade to the command above:

python3 -m pip install --upgrade --user ansible



```
┌──(niprovip㉿kali)-[~]
└─$ python3 -m pip install --upgrade --user ansible
Requirement already satisfied: ansible in /usr/lib/python3/dist-packages (7.7.0)
Collecting ansible
  Downloading ansible-8.6.0-py3-none-any.whl.metadata (7.9 kB)
Collecting ansible-core~=2.15.6 (from ansible)
  Downloading ansible_core-2.15.6-py3-none-any.whl.metadata (7.0 kB)
Requirement already satisfied: jinja2≥3.0.0 in /usr/lib/python3/dist-packages (from ansible-core~=2.15.6→ansible) (3.1.2)
Requirement already satisfied: PyYAML≥5.1 in /usr/lib/python3/dist-packages (from ansible-core~=2.15.6→ansible) (6.0.1)
Requirement already satisfied: cryptography in /usr/lib/python3/dist-packages (from ansible-core~=2.15.6→ansible) (38.0.4)
Requirement already satisfied: packaging in /usr/lib/python3/dist-packages (from ansible-core~=2.15.6→ansible) (23.1)
Requirement already satisfied: resolvelib<1.1.0,≥0.5.3 in /usr/lib/python3/dist-packages (from ansible-core~=2.15.6→ansible) (1.0.1)
Downloading ansible-8.6.0-py3-none-any.whl (48.3 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 48.3/48.3 MB 288.4 kB/s eta 0:00:00
Downloading ansible_core-2.15.6-py3-none-any.whl (2.2 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 2.2/2.2 MB 350.1 kB/s eta 0:00:00
Installing collected packages: ansible-core, ansible
  WARNING: The scripts ansible, ansible-config, ansible-connection, ansible-console, ansible-doc, ansible-galaxy, ansible-inventory, ansible-playbook, ansible-pull and ansible-vault are installed in '/home/niprovip/.local/bin' which is
 not on PATH.
  Consider adding this directory to PATH or, if you prefer to suppress this warning, use --no-warn-script-location.
  WARNING: The script ansible-community is installed in '/home/niprovip/.local/bin' which is not on PATH.
  Consider adding this directory to PATH or, if you prefer to suppress this warning, use --no-warn-script-location.
Successfully installed ansible-8.6.0 ansible-core-2.15.6
```

- Check the ansible version again after installation and upgradation with the ***anssible --version*** command. We have the version results below.



```
┌──(niprovip㉿kali)-[~]
└─$ ansible --version
ansible [core 2.15.6]
  config file = None
  configured module search path = ['/home/niprovip/.ansible/plugins/modules', '/usr/share/ansible/plugins/modules']
  ansible python module location = /home/niprovip/.local/lib/python3.11/site-packages/ansible
  ansible collection location = /home/niprovip/.ansible/collections:/usr/share/ansible/collections
  executable location = /usr/bin/ansible
  python version = 3.11.6 (main, Oct  8 2023, 05:06:43) [GCC 13.2.0] (/usr/bin/python3)
  jinja version = 3.1.2
  libyaml = True
```

B. Configure SSH Key and declare inventory file

- Ansible operates on an agentless mechanism, meaning there is no need to install an agent on client machines for control, instead ansible will use control of clients via SSH. Therefore, at this step we can use 2 ways for Ansible to control client machines.

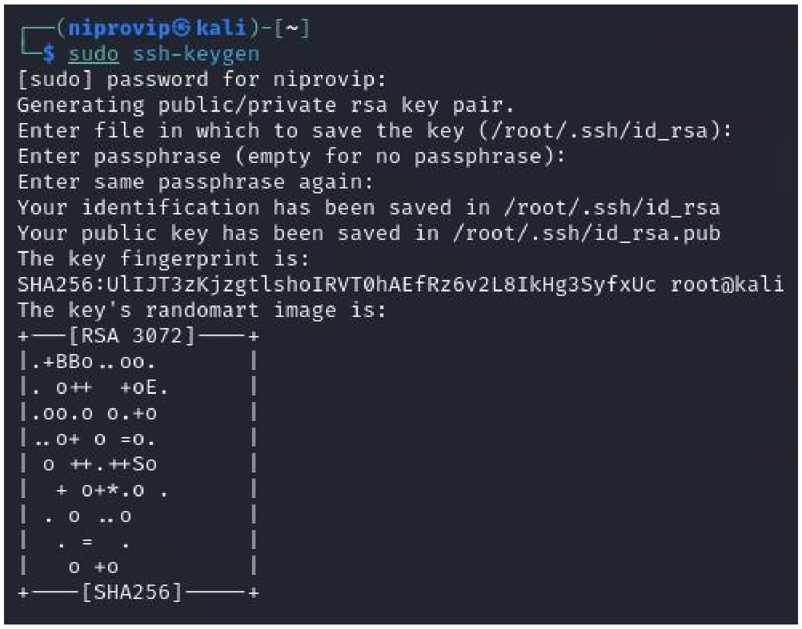
+ **Method 1:** Use ssh usename and port to declare in inventory. These methods are not recommended when used in practice because the clear text password will be displayed, or if using this method, it is necessary to secure this inventory file with ansible-vault.

+ **Method 2**: Use ssh keypair. This means we will generate private keys and public keys on the AnisbleServer node and copy them to client nodes (also known as hosts).

B.1. Generate SSH keys for nodes

- At the AnsibleServer node, create an SSH Key, then copy the keys to the other node. The goal is to use keypair so we don't have to enter a password every time we log in to clients.

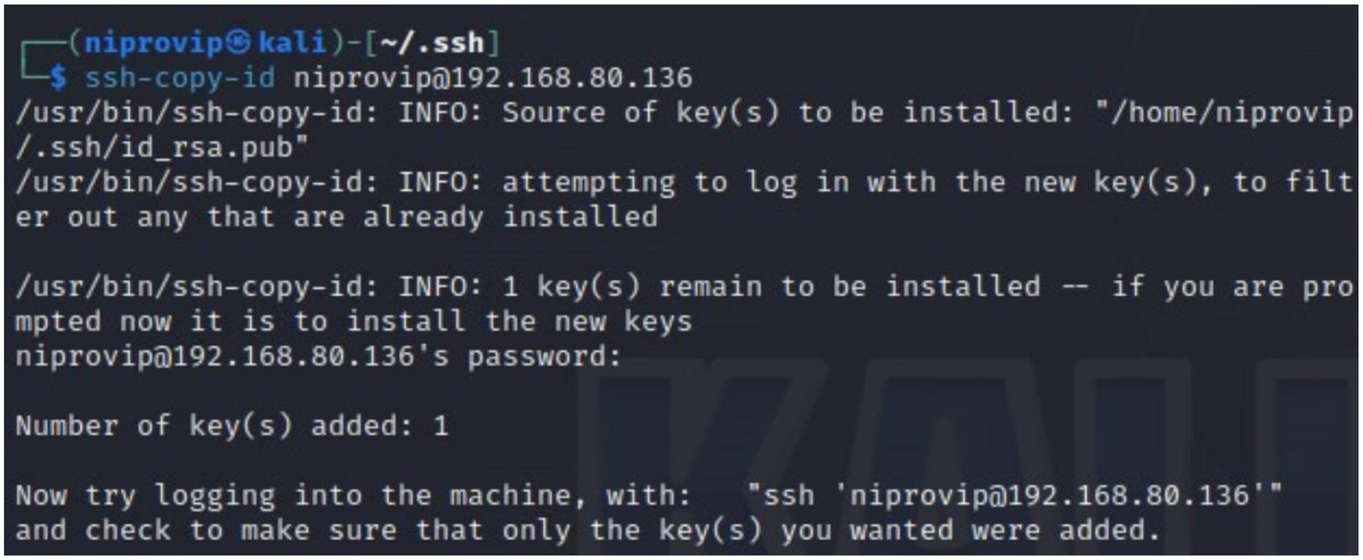
- Stand at the root user of the AnsibleServer node and perform the key generation step: ssh-keygen. Perform Enter operations and leave the options as default:



```
┌──(niprovip㉿kali)-[~]
└─$ sudo ssh-keygen
[sudo] password for niprovip:
Generating public/private rsa key pair.
Enter file in which to save the key (/root/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /root/.ssh/id_rsa
Your public key has been saved in /root/.ssh/id_rsa.pub
The key fingerprint is:
SHA256:UlIJT3zKjzgtlshoIRVT0hAEfRz6v2L8IkHg3SyfxUc root@kali
The key's randomart image is:
+---[RSA 3072]----+
|.+BBo..oo.       |
|. o++  +oE.      |
|.oo.o o.+o       |
|..o+ o =o.       |
| o ++.++So       |
|  + o+*.o .      |
| . o ..o         |
|  . =  .         |
|   o +o          |
+----[SHA256]-----+
```

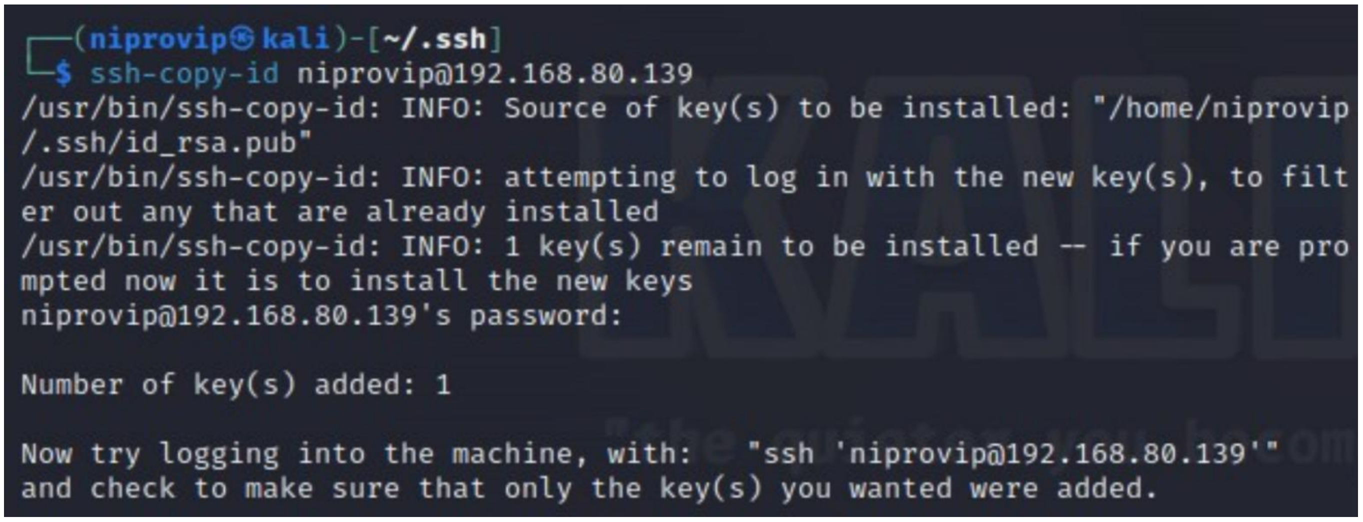
- Copy the key file to the remaining nodes

+ Copy key to node1 192.168.80.136



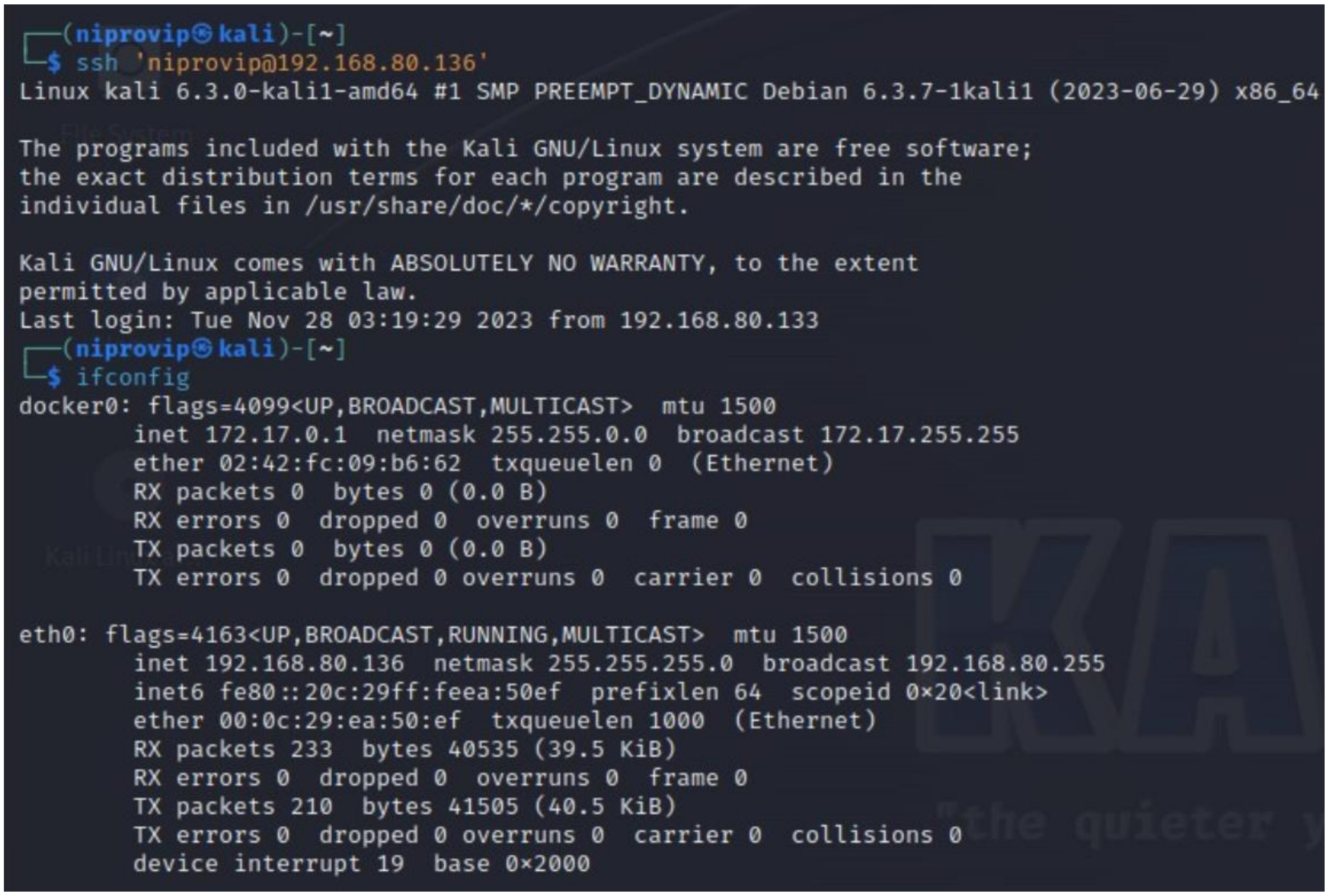
```
┌──(niprovip㉿kali)-[~/.ssh]
└─$ ssh-copy-id niprovip@192.168.80.136
/usr/bin/ssh-copy-id: INFO: Source of key(s) to be installed: "/home/niprovip
/.ssh/id_rsa.pub"
/usr/bin/ssh-copy-id: INFO: attempting to log in with the new key(s), to filt
er out any that are already installed

/usr/bin/ssh-copy-id: INFO: 1 key(s) remain to be installed -- if you are pro
mpted now it is to install the new keys
niprovip@192.168.80.136's password:

Number of key(s) added: 1

Now try logging into the machine, with:   "ssh 'niprovip@192.168.80.136'"
and check to make sure that only the key(s) you wanted were added.
```

+ Do the same with node2 192.168.80.139

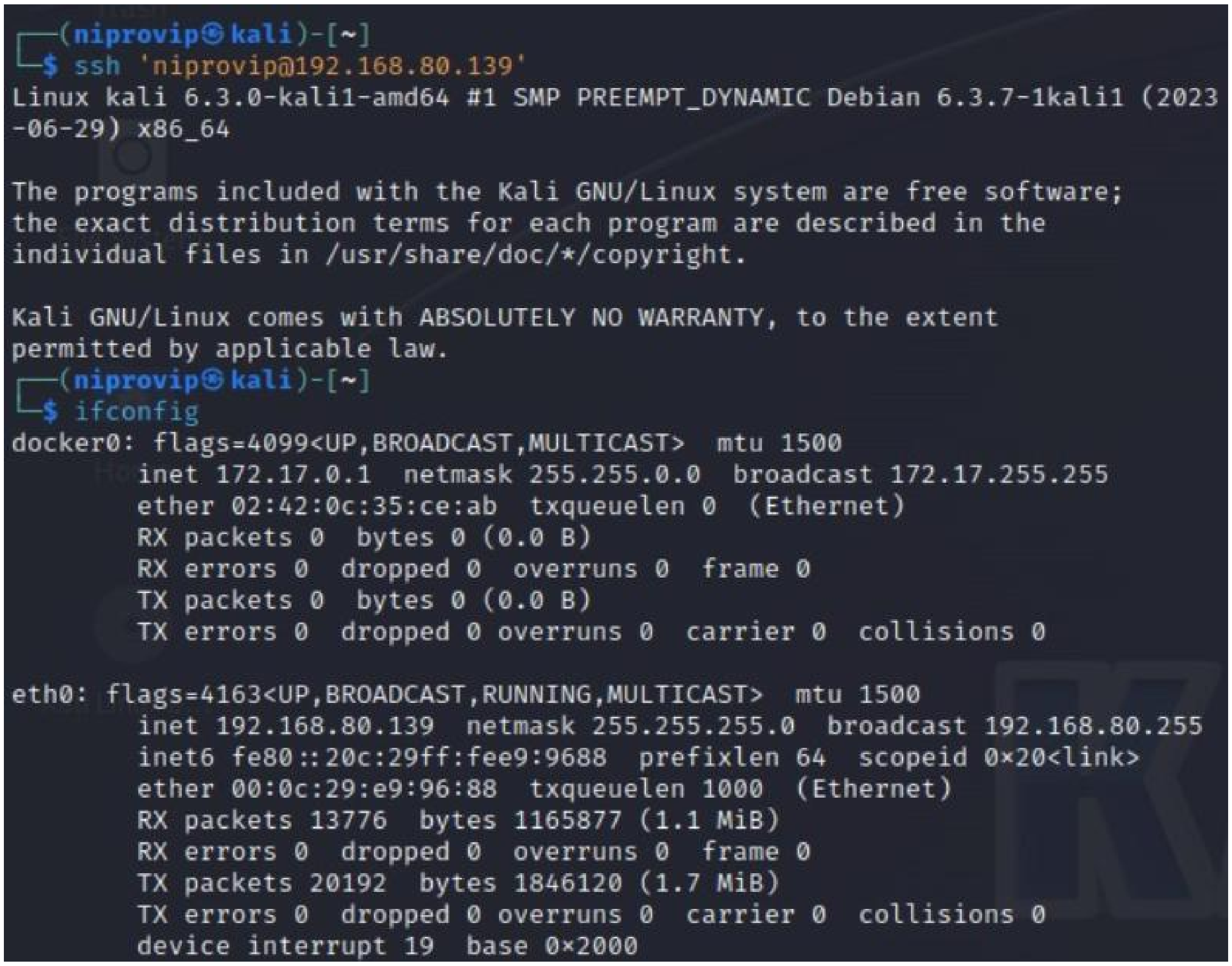


```
┌──(niprovip㉿kali)-[~/.ssh]
└─$ ssh-copy-id niprovip@192.168.80.139
/usr/bin/ssh-copy-id: INFO: Source of key(s) to be installed: "/home/niprovip
/.ssh/id_rsa.pub"
/usr/bin/ssh-copy-id: INFO: attempting to log in with the new key(s), to filt
er out any that are already installed
/usr/bin/ssh-copy-id: INFO: 1 key(s) remain to be installed -- if you are pro
mpted now it is to install the new keys
niprovip@192.168.80.139's password:

Number of key(s) added: 1

Now try logging into the machine, with:   "ssh 'niprovip@192.168.80.139'"
and check to make sure that only the key(s) you wanted were added.
```

- Then from the AnsibleServer node, try ssh to the client1 & client2 nodes. If we are not asked for a password, we have successfully used the ssh key.



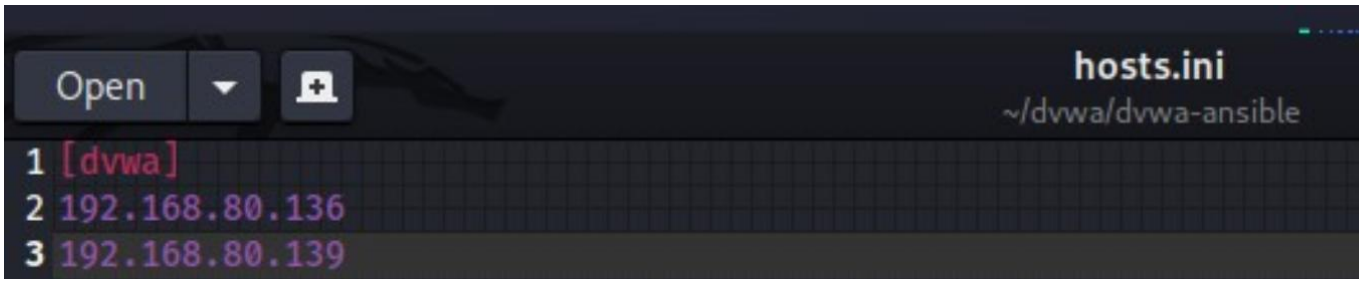
```
┌──(niprovip㉿kali)-[~]
└─$ ssh 'niprovip@192.168.80.136'
Linux kali 6.3.0-kali1-amd64 #1 SMP PREEMPT_DYNAMIC Debian 6.3.7-1kali1 (2023-06-29) x86_64

The programs included with the Kali GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Kali GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Tue Nov 28 03:19:29 2023 from 192.168.80.133
┌──(niprovip㉿kali)-[~]
└─$ ifconfig
docker0: flags=4099<UP,BROADCAST,MULTICAST>  mtu 1500
        inet 172.17.0.1  netmask 255.255.0.0  broadcast 172.17.255.255
        ether 02:42:fc:09:b6:62  txqueuelen 0  (Ethernet)
        RX packets 0  bytes 0 (0.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 0  bytes 0 (0.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.80.136  netmask 255.255.255.0  broadcast 192.168.80.255
        inet6 fe80::20c:29ff:feea:50ef  prefixlen 64  scopeid 0x20<link>
        ether 00:0c:29:ea:50:ef  txqueuelen 1000  (Ethernet)
        RX packets 233  bytes 40535 (39.5 KiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 210  bytes 41505 (40.5 KiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
        device interrupt 19  base 0x2000
```



```
┌──(niprovip㉿kali)-[~]
└─$ ssh 'niprovip@192.168.80.139'
Linux kali 6.3.0-kali1-amd64 #1 SMP PREEMPT_DYNAMIC Debian 6.3.7-1kali1 (2023
-06-29) x86_64

The programs included with the Kali GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Kali GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
┌──(niprovip㉿kali)-[~]
└─$ ifconfig
docker0: flags=4099<UP,BROADCAST,MULTICAST>  mtu 1500
        inet 172.17.0.1  netmask 255.255.0.0  broadcast 172.17.255.255
        ether 02:42:0c:35:ce:ab  txqueuelen 0  (Ethernet)
        RX packets 0  bytes 0 (0.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 0  bytes 0 (0.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.80.139  netmask 255.255.255.0  broadcast 192.168.80.255
        inet6 fe80::20c:29ff:fee9:9688  prefixlen 64  scopeid 0x20<link>
        ether 00:0c:29:e9:96:88  txqueuelen 1000  (Ethernet)
        RX packets 13776  bytes 1165877 (1.1 MiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 20192  bytes 1846120 (1.7 MiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
        device interrupt 19  base 0x2000
```

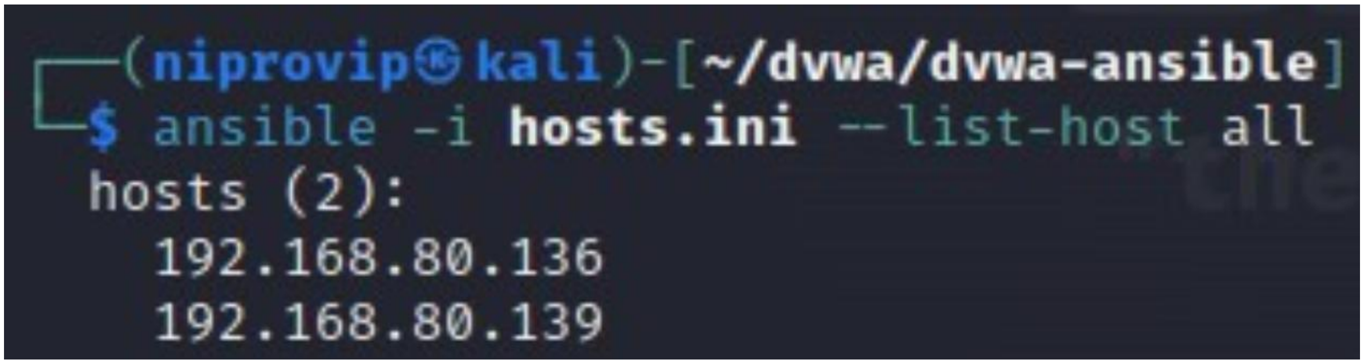
B.2 Declare inventory file

- Create a new inventory file .ini to declare the hosts that need to be managed



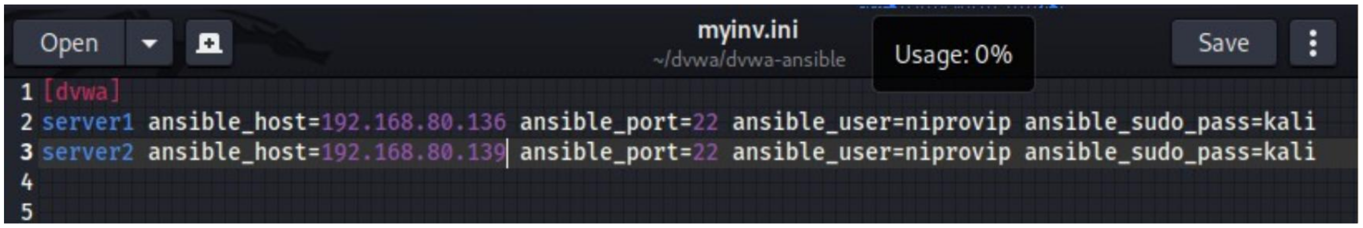
```
[dvwa]
192.168.80.136
192.168.80.139
```

- Check the host list again by running the command:



```
┌──(niprovip㉿kali)-[~/dvwa/dvwa-ansible]
└─$ ansible -i hosts.ini --list-host all
  hosts (2):
    192.168.80.136
    192.168.80.139
```

- In reality, we need to declare additional options about passwords, ports, and even users that AnsibleServer is allowed to use to control hosts. A relatively complete inventory file will have the following format:



```
[dvwa]
server1 ansible_host=192.168.80.136 ansible_port=22 ansible_user=niprovip ansible_sudo_pass=kali
server2 ansible_host=192.168.80.139 ansible_port=22 ansible_user=niprovip ansible_sudo_pass=kali
```

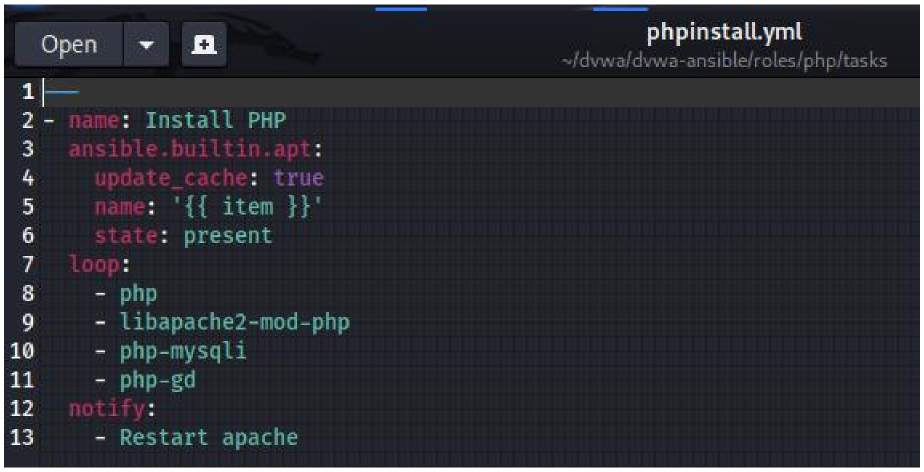
server1, server2: Correspondingly are the hostnames of the nodes

ansible_host: IP address of the corresponding client node.

ansible_port: Port of SSH client side, if we change it, we will adjust it correctly.

ansible_user: Is the username of the client that AnsibleServer will use to interact. In the above step we used the root user and passed the SSH Key.

C. Install PHP



```
---
- name: Install PHP
  ansible.builtin.apt:
    update_cache: true
    name: '{{ item }}'
    state: present
  loop:
    - php
    - libapache2-mod-php
    - php-mysqli
    - php-gd
  notify:
    - Restart apache
```

phpinstall.yml

The module used in this task is ansible.builtin.apt. This module is part of the Ansible core collection, which provides modules for managing packages on Debian-based systems.

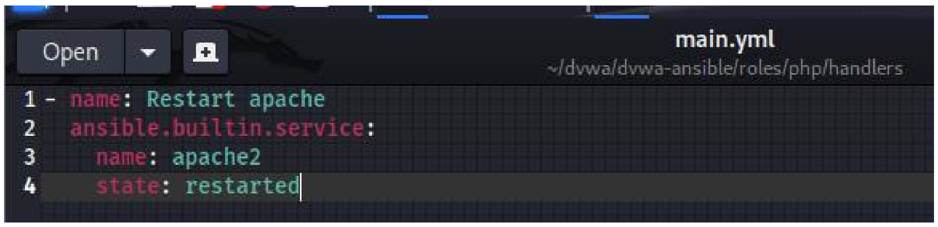
update_cache: This parameter specifies whether to update the package cache before installing the packages. The value is true, which means that the cache will be updated.

name: This parameter specifies the name of the package to install. The value is an Ansible variable named {{ item }}. This variable is expected to be a list of package names.

state: This parameter specifies the desired state of the package. The value is present, which means that the package will be installed if it is not already installed.

loop: This parameter specifies that the task should be looped over the value of the name parameter. This means that the task will be executed once for each package name in the list.

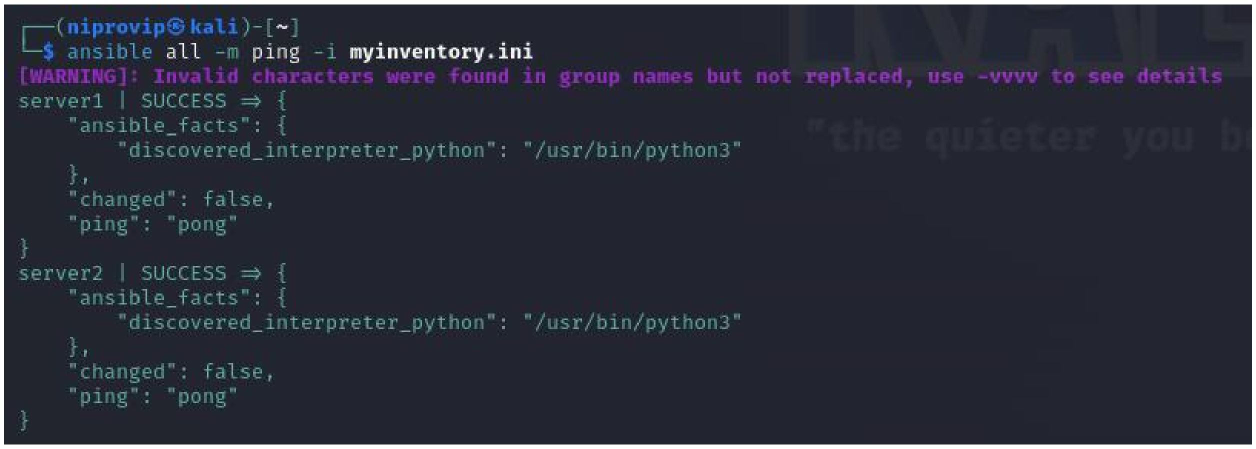
notify: This parameter specifies a list of handlers to be notified when the task is completed.



```
- name: Restart apache
  ansible.builtin.service:
    name: apache2
    state: restarted
```

D.Use some basic test commands

- To check if the above declaration is correct and can proceed with the next steps, we will use the ping module with the -m option.

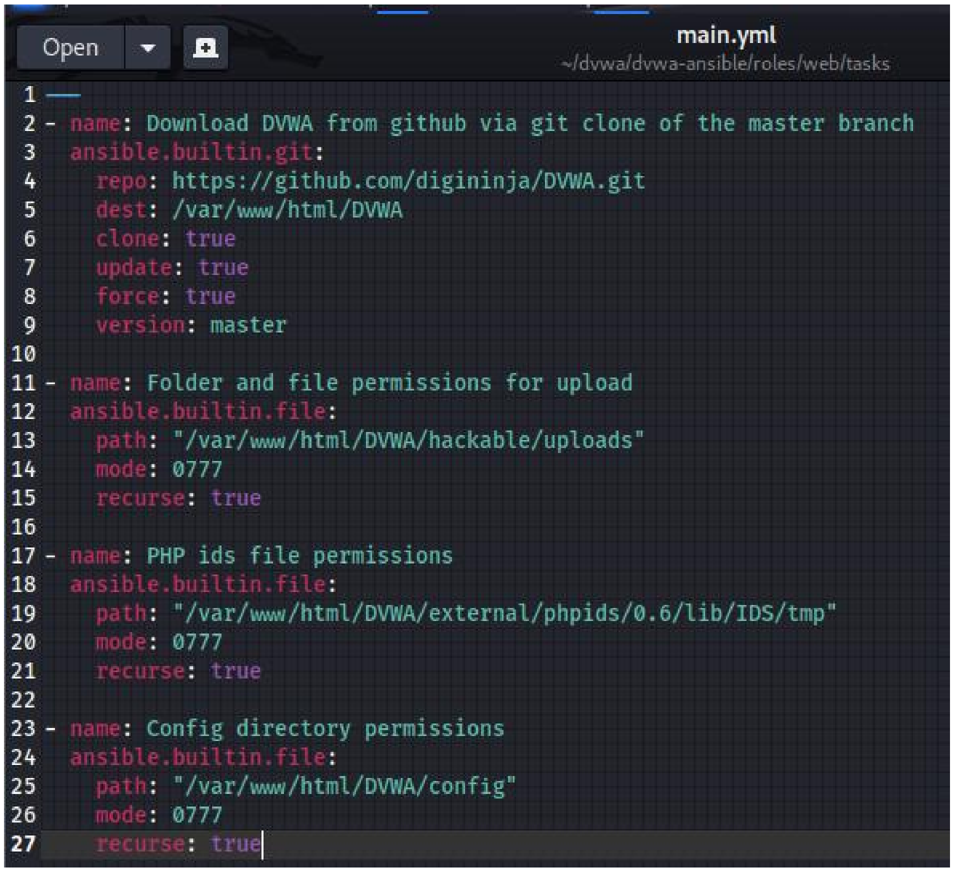


```
┌──(niprovip㉿kali)-[~]
└─$ ansible all -m ping -i myinventory.ini
[WARNING]: Invalid characters were found in group names but not replaced, use -vvvv to see details
server1 | SUCCESS => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python3"
    },
    "changed": false,
    "ping": "pong"
}
server2 | SUCCESS => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python3"
    },
    "changed": false,
    "ping": "pong"
}
```

- In the next part, Configuration, we will write a playbook to automatically install software/services on corresponding hosts.

2.3. Configuration

A. Configure DVWA



```yaml
---
- name: Download DVWA from github via git clone of the master branch
  ansible.builtin.git:
    repo: https://github.com/digininja/DVWA.git
    dest: /var/www/html/DVWA
    clone: true
    update: true
    force: true
    version: master

- name: Folder and file permissions for upload
  ansible.builtin.file:
    path: "/var/www/html/DVWA/hackable/uploads"
    mode: 0777
    recurse: true

- name: PHP ids file permissions
  ansible.builtin.file:
    path: "/var/www/html/DVWA/external/phpids/0.6/lib/IDS/tmp"
    mode: 0777
    recurse: true

- name: Config directory permissions
  ansible.builtin.file:
    path: "/var/www/html/DVWA/config"
    mode: 0777
    recurse: true
```

Task 1: Download DVWA from GitHub

This task uses the ansible.builtin.git module to clone the DVWA repository from GitHub. The repository is cloned into the /var/www/html/DVWA directory. The following options are used:

- *repo*: This parameter specifies the URL of the repository to clone. In this case, the value is https://github.com/digininja/DVWA.git.

- *dest*: This parameter specifies the directory where the repository should be cloned. In this case, the value is /var/www/html/DVWA.

- *clone*: This parameter specifies whether to clone the repository. In this case, the value is true, which means that the repository will be cloned.

- *update*: This parameter specifies whether to update the repository if it is already cloned. In this case, the value is true, which means that the repository will be updated if it is already cloned.

- *force*: This parameter specifies whether to force the clone. In this case, the value is true, which means that the clone will be forced even if the directory already exists.

- *version*: This parameter specifies the version of the repository to clone. In this case, the value is master, which means that the master branch will be cloned.

Task 2: Set folder and file permissions for upload

This task uses the ansible.builtin.file module to set the permissions of the /var/www/html/DVWA/hackable/uploads directory and its contents to 0777. This means that the directory and its contents will be readable, writable, and executable by all users.

- path: This parameter specifies the path to the file or directory to modify. In this case, the value is /var/www/html/DVWA/hackable/uploads.

- mode: This parameter specifies the permissions of the file or directory. In this case, the value is 0777, which means that the file or directory will be readable, writable, and executable by all users.

- recurse: This parameter specifies whether to recursively modify the permissions of the directory and its contents. In this case, the value is true, which means that the permissions of the directory and its contents will be recursively modified.

Task 3: Set PHP IDS file permissions

This task uses the ansible.builtin.file module to set the permissions of the /var/www/html/DVWA/external/phpids/0.6/lib/IDS/tmp directory and its contents to 0777. This means that the directory and its contents will be readable, writable, and executable by all users.

- path: This parameter specifies the path to the file or directory to modify. In this case, the value is /var/www/html/DVWA/external/phpids/0.6/lib/IDS/tmp.

- mode: This parameter specifies the permissions of the file or directory. In this case, the value is 0777, which means that the file or directory will be readable, writable, and executable by all users.

- recurse: This parameter specifies whether to recursively modify the permissions of the directory and its contents. In this case, the value is true, which means that the permissions of the directory and its contents will be recursively modified.

Task 4: Set config directory permissions

This task uses the ansible.builtin.file module to set the permissions of the /var/www/html/DVWA/config directory and its contents to 0777. This means that the directory and its contents will be readable, writable, and executable by all users.

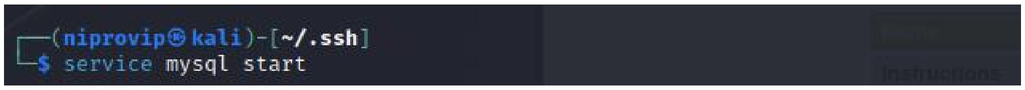
- path: This parameter specifies the path to the file or directory to modify. In this case, the value is /var/www/html/DVWA/config.

- mode: This parameter specifies the permissions of the file or directory. In this case, the value is 0777, which means that the file or directory will be readable, writable, and executable by all users.

- ***recurse***: This parameter specifies whether to recursively modify the permissions of the directory and its contents. In this case, the value is true, which means that the permissions of the directory and its contents will be recursively modified.

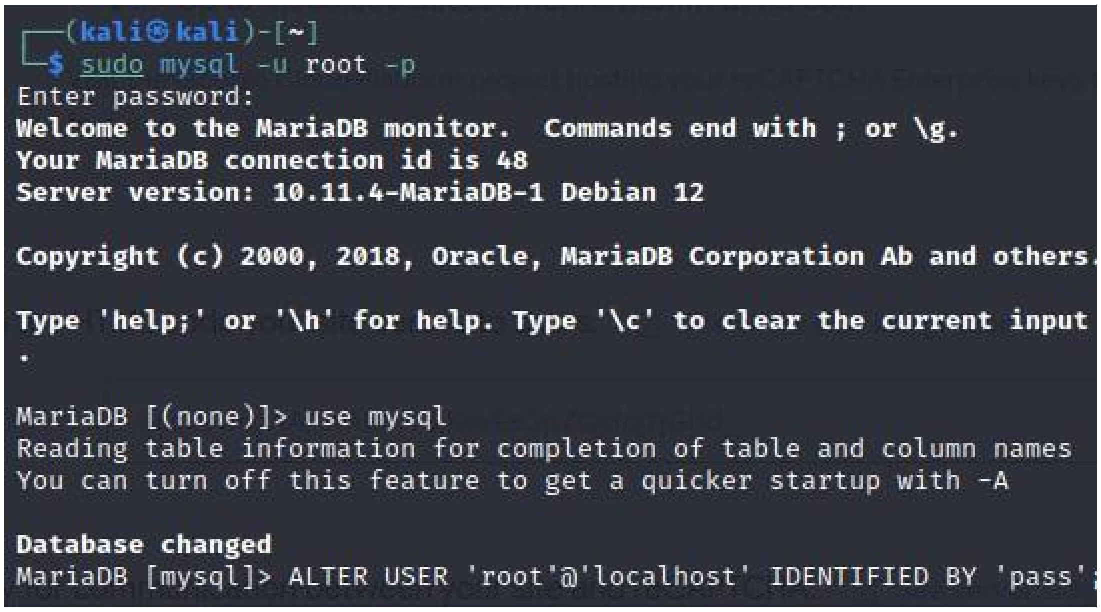
B.Configure MySQL Database

- Start the MySQL service with the command below:



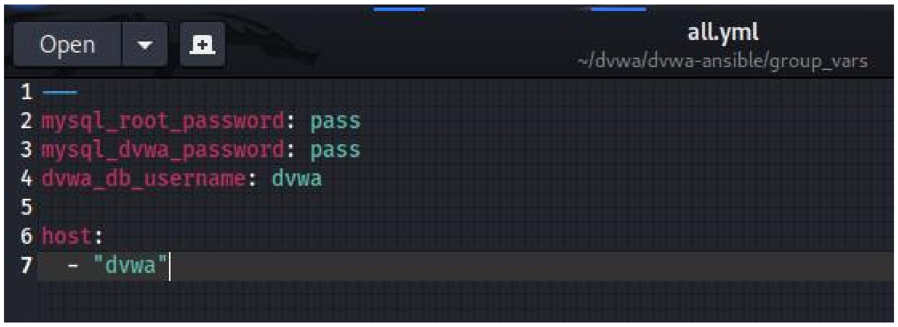
```
┌──(niprovip㉿kali)-[~/.ssh]
└─$ service mysql start
```

- Loging in the MySql with root account and then change its password:

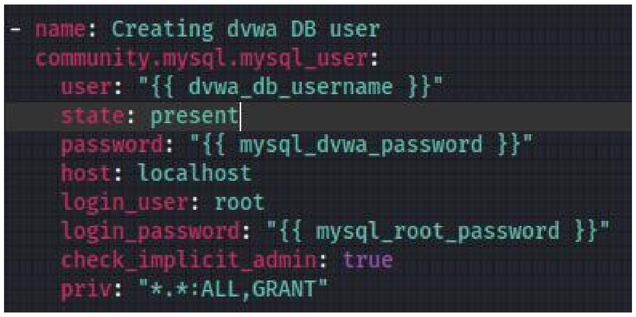


```
┌──(kali㉿kali)-[~]
└─$ sudo mysql -u root -p
Enter password:
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 48
Server version: 10.11.4-MariaDB-1 Debian 12

Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others

Type 'help;' or '\h' for help. Type '\c' to clear the current input
.

MariaDB [(none)]> use mysql
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
MariaDB [mysql]> ALTER USER 'root'@'localhost' IDENTIFIED BY 'pass'
```

- Declare vars for root user's password and create new user acount.



```
---
mysql_root_password: pass
mysql_dvwa_password: pass
dvwa_db_username: dvwa

host:
  - "dvwa"
```



```
- name: Creating dvwa DB user
  community.mysql.mysql_user:
    user: "{{ dvwa_db_username }}"
    state: present
    password: "{{ mysql_dvwa_password }}"
    host: localhost
    login_user: root
    login_password: "{{ mysql_root_password }}"
    check_implicit_admin: true
    priv: "*.*:ALL,GRANT"
```

The module used in this task is community.mysql.mysql_user. This module is part of the community.mysql collection of Ansible modules, which provides modules for managing MySQL databases.

The task has the following parameters:

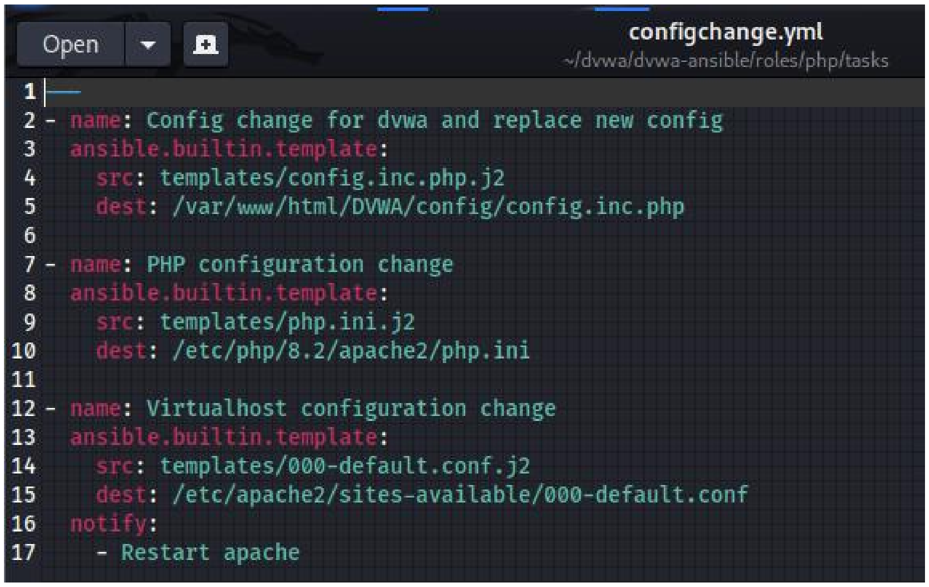
user: This parameter specifies the name of the MySQL user to create

password: This parameter specifies the password for the MySQL user

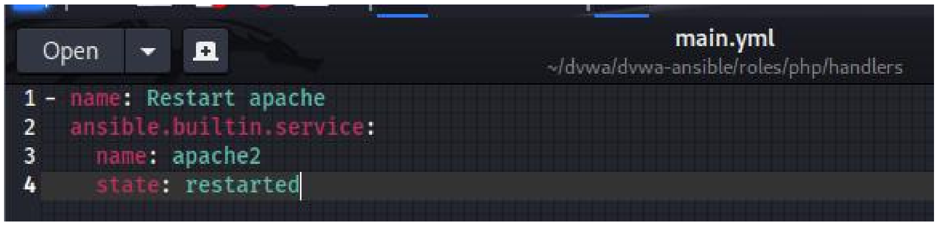
host: This parameter specifies the host that the MySQL user can connect from.

priv: This parameter specifies the privileges to grant to the MySQL user. In this case, the value is *.*:ALL,GRANT, which means that the MySQL user will have all privileges on all databases and will be able to grant privileges to other users.

C. Configure Apache Server



```yaml
---
- name: Config change for dvwa and replace new config
  ansible.builtin.template:
    src: templates/config.inc.php.j2
    dest: /var/www/html/DVWA/config/config.inc.php

- name: PHP configuration change
  ansible.builtin.template:
    src: templates/php.ini.j2
    dest: /etc/php/8.2/apache2/php.ini

- name: Virtualhost configuration change
  ansible.builtin.template:
    src: templates/000-default.conf.j2
    dest: /etc/apache2/sites-available/000-default.conf
  notify:
    - Restart apache
```



```yaml
- name: Restart apache
  ansible.builtin.service:
    name: apache2
    state: restarted
```

Task 1: Replace DVWA configuration file

This task uses the ansible.builtin.template module to replace the /var/www/html/DVWA/config/config.inc.php file with a new configuration file generated from the templates/config.inc.php.j2 template. The j2 extension indicates that the template is a Jinja2 template.

Task 2: Modify PHP configuration file

This task uses the ansible.builtin.template module to modify the /etc/php/8.2/apache2/php.ini file with settings from the templates/php.ini.j2 template.

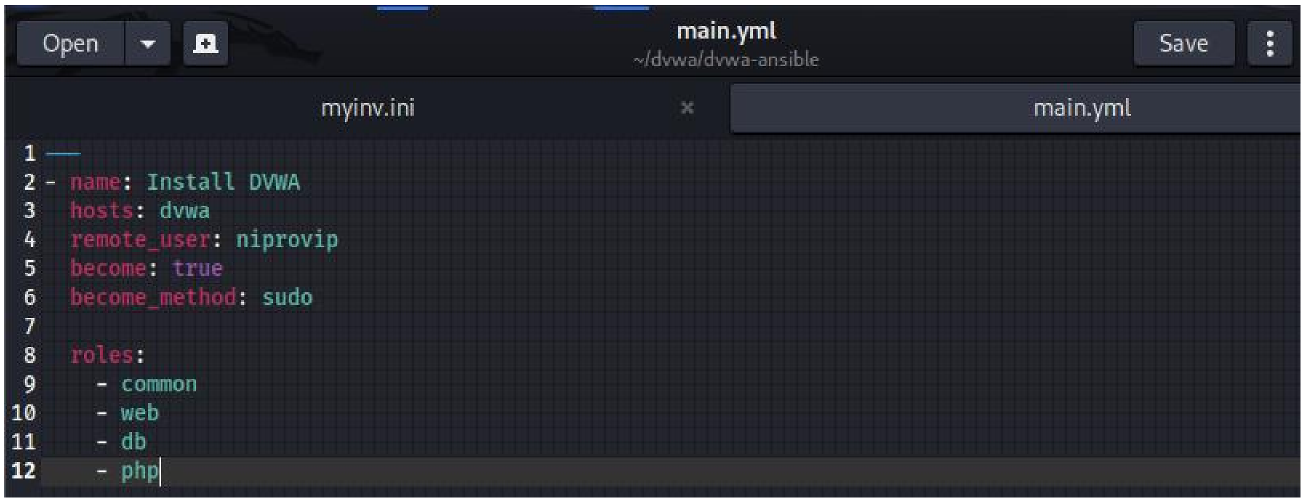
Task 3: Modify Apache virtual host configuration file

This task uses the ansible.builtin.template module to modify the /etc/apache2/sites-available/000-default.conf file with settings from the templates/000-default.conf.j2 template.

III. Result and Conclusion

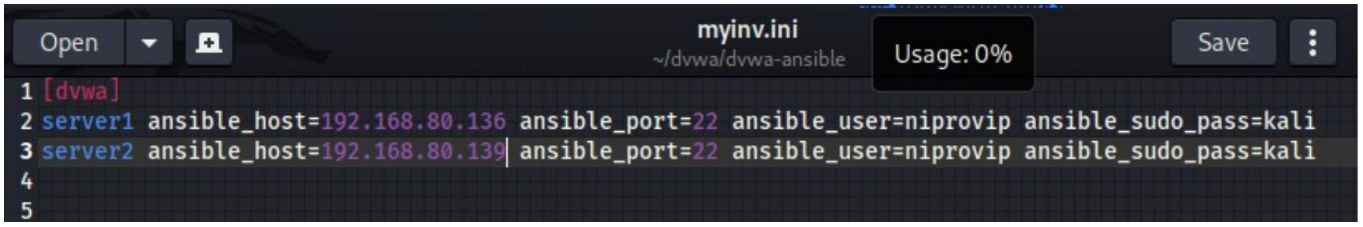
3.1. Result

- File main.yml. The roles parameter specifies a list of roles to apply to the hosts. In this case, the value is a list containing the following roles: common, web, db, php



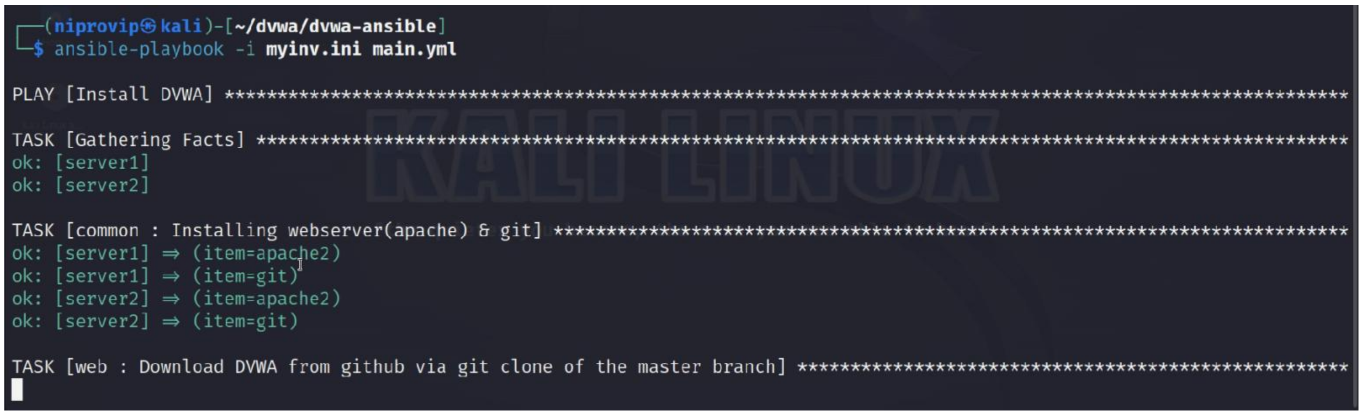
```
---
- name: Install DVWA
  hosts: dvwa
  remote_user: niprovip
  become: true
  become_method: sudo

  roles:
    - common
    - web
    - db
    - php
```

-File inventory.ini



```
[dvwa]
server1 ansible_host=192.168.80.136 ansible_port=22 ansible_user=niprovip ansible_sudo_pass=kali
server2 ansible_host=192.168.80.139 ansible_port=22 ansible_user=niprovip ansible_sudo_pass=kali
```

- Run the main.yml playbook with ansible-playbook

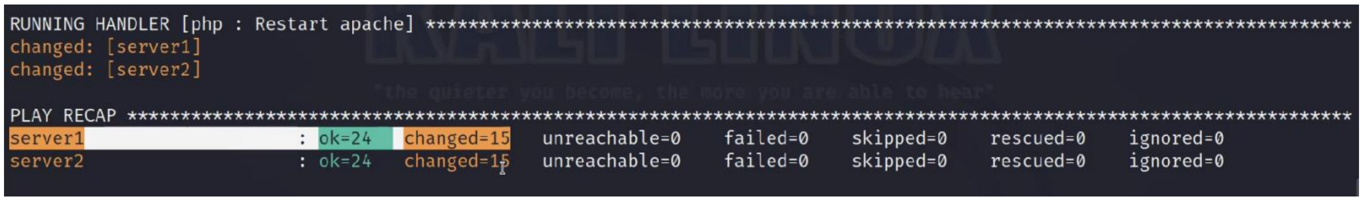


```
┌──(niprovip㉿kali)-[~/dvwa/dvwa-ansible]
└─$ ansible-playbook -i myinv.ini main.yml

PLAY [Install DVWA] ***************************************************************************************

TASK [Gathering Facts] ************************************************************************************
ok: [server1]
ok: [server2]

TASK [common : Installing webserver(apache) & git] ********************************************************
ok: [server1] => (item=apache2)
ok: [server1] => (item=git)
ok: [server2] => (item=apache2)
ok: [server2] => (item=git)

TASK [web : Download DVWA from github via git clone of the master branch] *********************************
```

…

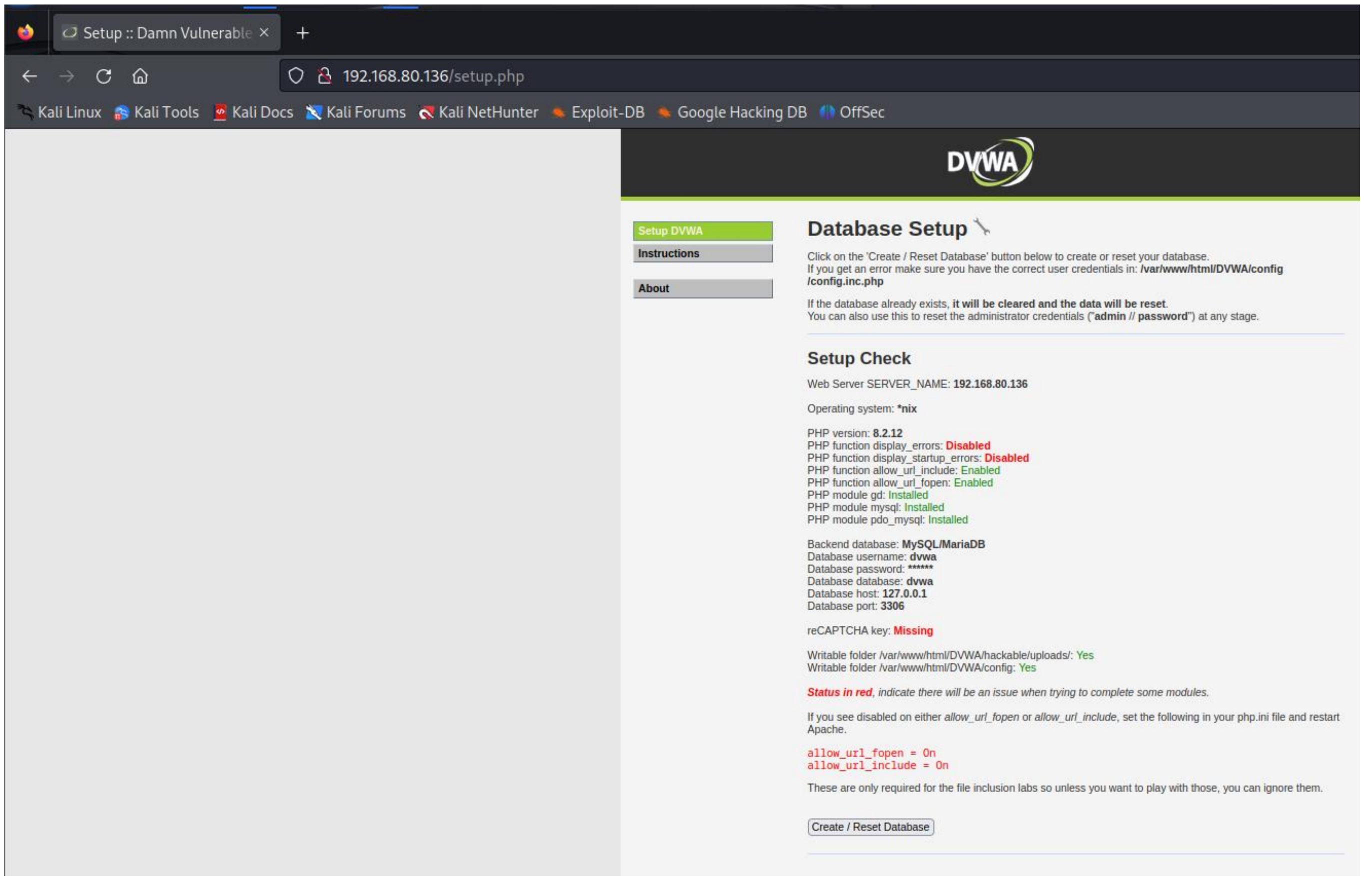
…

…



```
RUNNING HANDLER [php : Restart apache] ******************************************************************************
changed: [server1]
changed: [server2]

PLAY RECAP **********************************************************************************************************
server1                    : ok=24   changed=15   unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
server2                    : ok=24   changed=15   unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
```

- Open the host's browser and enter the URL 192.168.80.136/setup.php. That will open the setup.php web page as shown in the image below:



Database Setup

Click on the 'Create / Reset Database' button below to create or reset your database.
If you get an error make sure you have the correct user credentials in: **/var/www/html/DVWA/config/config.inc.php**

If the database already exists, **it will be cleared and the data will be reset**.
You can also use this to reset the administrator credentials ("**admin** // **password**") at any stage.

Setup Check

Web Server SERVER_NAME: **192.168.80.136**

Operating system: ***nix**

PHP version: **8.2.12**
PHP function display_errors: **Disabled**
PHP function display_startup_errors: **Disabled**
PHP function allow_url_include: Enabled
PHP function allow_url_fopen: Enabled
PHP module gd: Installed
PHP module mysql: Installed
PHP module pdo_mysql: Installed

Backend database: **MySQL/MariaDB**
Database username: **dvwa**
Database password: **********
Database database: **dvwa**
Database host: **127.0.0.1**
Database port: **3306**

reCAPTCHA key: **Missing**

Writable folder /var/www/html/DVWA/hackable/uploads/: Yes
Writable folder /var/www/html/DVWA/config: Yes

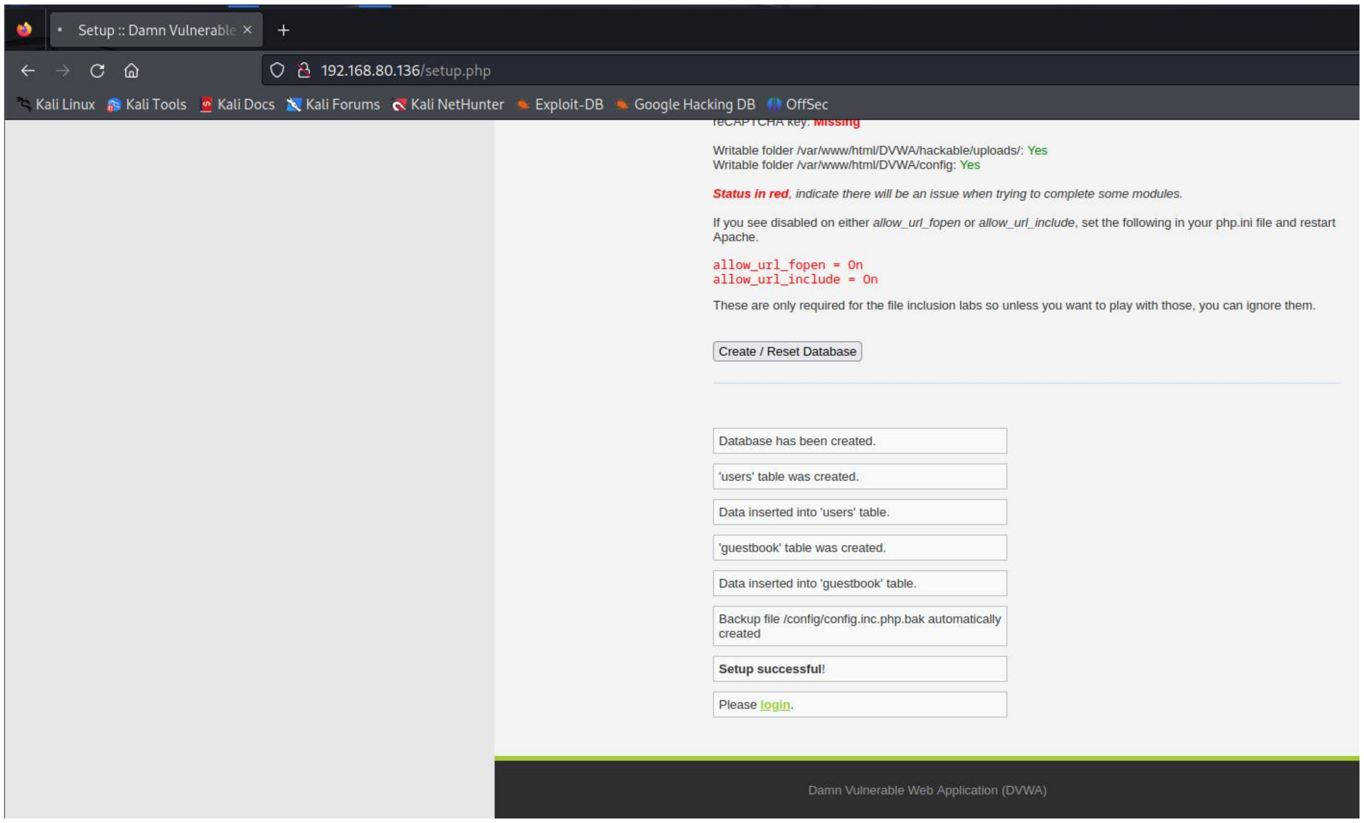
Status in red*, indicate there will be an issue when trying to complete some modules.*

If you see disabled on either *allow_url_fopen* or *allow_url_include*, set the following in your php.ini file and restart Apache.

```
allow_url_fopen = On
allow_url_include = On
```

These are only required for the file inclusion labs so unless you want to play with those, you can ignore them.

- Click the Create / Reset Database button.


Writable folder /var/www/html/DVWA/hackable/uploads/: Yes
Writable folder /var/www/html/DVWA/config: Yes

Status in red, indicate there will be an issue when trying to complete some modules.

If you see disabled on either *allow_url_fopen* or *allow_url_include*, set the following in your php.ini file and restart Apache.

```
allow_url_fopen = On
allow_url_include = On
```

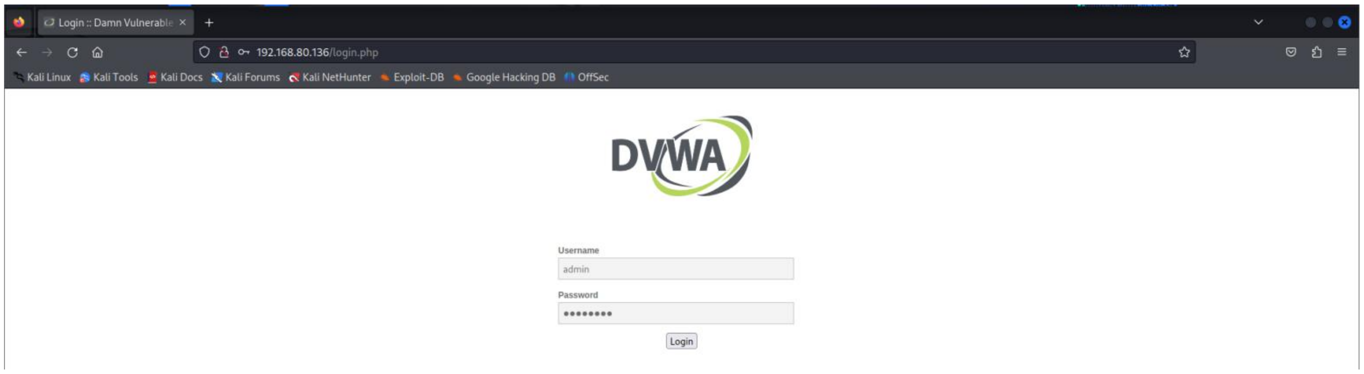
These are only required for the file inclusion labs so unless you want to play with those, you can ignore them.

Database has been created.
'users' table was created.
Data inserted into 'users' table.
'guestbook' table was created.
Data inserted into 'guestbook' table.
Backup file /config/config.inc.php.bak automatically created
Setup successful!
Please login.

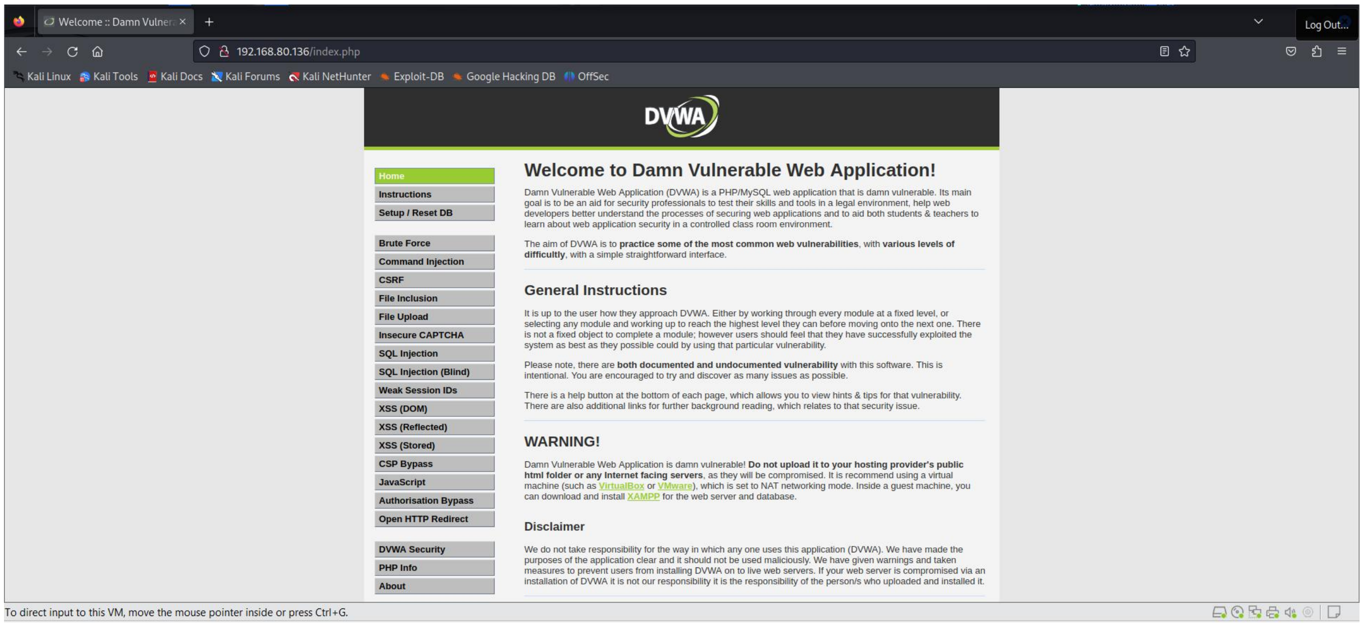
- After some time, we will be redirected to the DVWA login page. Log in with these default creadentials

+ Username - admin

+ Password - password



- Once logged in, we will see the DVWA main page:



Welcome to Damn Vulnerable Web Application!

Damn Vulnerable Web Application (DVWA) is a PHP/MySQL web application that is damn vulnerable. Its main goal is to be an aid for security professionals to test their skills and tools in a legal environment, help web developers better understand the processes of securing web applications and to aid both students & teachers to learn about web application security in a controlled class room environment.

The aim of DVWA is to **practice some of the most common web vulnerabilities**, with **various levels of difficultly**, with a simple straightforward interface.

General Instructions

It is up to the user how they approach DVWA. Either by working through every module at a fixed level, or selecting any module and working up to reach the highest level they can before moving onto the next one. There is not a fixed object to complete a module; however users should feel that they have successfully exploited the system as best as they possible could by using that particular vulnerability.

Please note, there are **both documented and undocumented vulnerability** with this software. This is intentional. You are encouraged to try and discover as many issues as possible.

There is a help button at the bottom of each page, which allows you to view hints & tips for that vulnerability. There are also additional links for further background reading, which relates to that security issue.

WARNING!

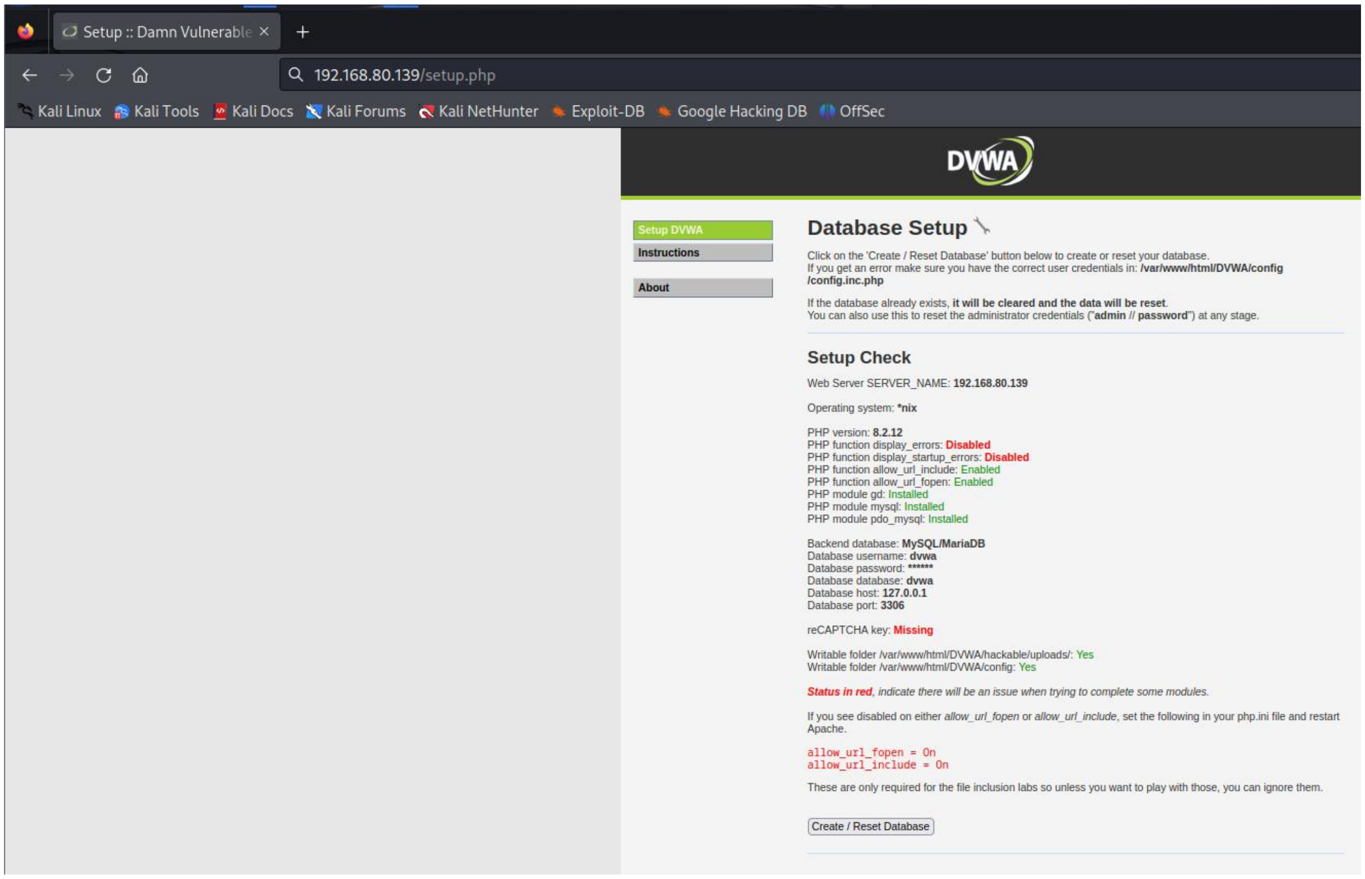
Damn Vulnerable Web Application is damn vulnerable! **Do not upload it to your hosting provider's public html folder or any Internet facing servers**, as they will be compromised. It is recommend using a virtual machine (such as VirtualBox or VMware), which is set to NAT networking mode. Inside a guest machine, you can download and install XAMPP for the web server and database.

Disclaimer

We do not take responsibility for the way in which any one uses this application (DVWA). We have made the purposes of the application clear and it should not be used maliciously. We have given warnings and taken measures to prevent users from installing DVWA on to live web servers. If your web server is compromised via an installation of DVWA it is not our responsibility it is the responsibility of the person/s who uploaded and installed it.

To direct input to this VM, move the mouse pointer inside or press Ctrl+G.

- Do the same with host 192.168.80.139



Database Setup

Click on the 'Create / Reset Database' button below to create or reset your database.
If you get an error make sure you have the correct user credentials in: **/var/www/html/DVWA/config/config.inc.php**

If the database already exists, **it will be cleared and the data will be reset**.
You can also use this to reset the administrator credentials ("**admin** // **password**") at any stage.

Setup Check

Web Server SERVER_NAME: **192.168.80.139**

Operating system: ***nix**

PHP version: **8.2.12**
PHP function display_errors: **Disabled**
PHP function display_startup_errors: **Disabled**
PHP function allow_url_include: Enabled
PHP function allow_url_fopen: Enabled
PHP module gd: Installed
PHP module mysql: Installed
PHP module pdo_mysql: Installed

Backend database: **MySQL/MariaDB**
Database username: **dvwa**
Database password: **********
Database database: **dvwa**
Database host: **127.0.0.1**
Database port: **3306**

reCAPTCHA key: **Missing**

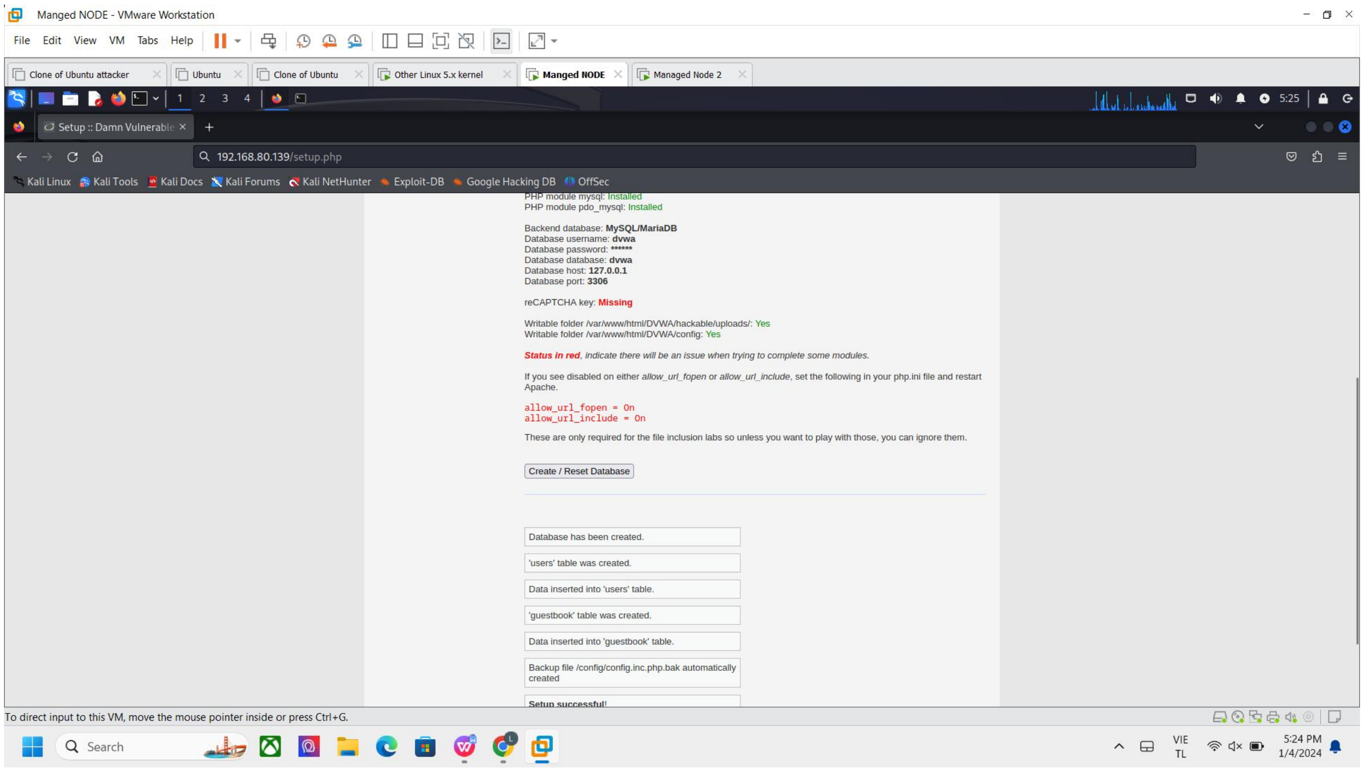
Writable folder /var/www/html/DVWA/hackable/uploads/: Yes
Writable folder /var/www/html/DVWA/config: Yes

Status in red*, indicate there will be an issue when trying to complete some modules.*

If you see disabled on either *allow_url_fopen* or *allow_url_include*, set the following in your php.ini file and restart Apache.

```
allow_url_fopen = On
allow_url_include = On
```

These are only required for the file inclusion labs so unless you want to play with those, you can ignore them.


Manged NODE - VMware Workstation

192.168.80.139/setup.php

PHP module mysql: Installed
PHP module pdo_mysql: Installed

Backend database: **MySQL/MariaDB**
Database username: **dvwa**
Database password: **********
Database database: **dvwa**
Database host: **127.0.0.1**
Database port: **3306**

reCAPTCHA key: **Missing**

Writable folder /var/www/html/DVWA/hackable/uploads/: Yes
Writable folder /var/www/html/DVWA/config: Yes

***Status in red**, indicate there will be an issue when trying to complete some modules.*

If you see disabled on either *allow_url_fopen* or *allow_url_include*, set the following in your php.ini file and restart Apache.

```
allow_url_fopen = On
allow_url_include = On
```

These are only required for the file inclusion labs so unless you want to play with those, you can ignore them.

Create / Reset Database

Database has been created.

'users' table was created.

Data inserted into 'users' table.

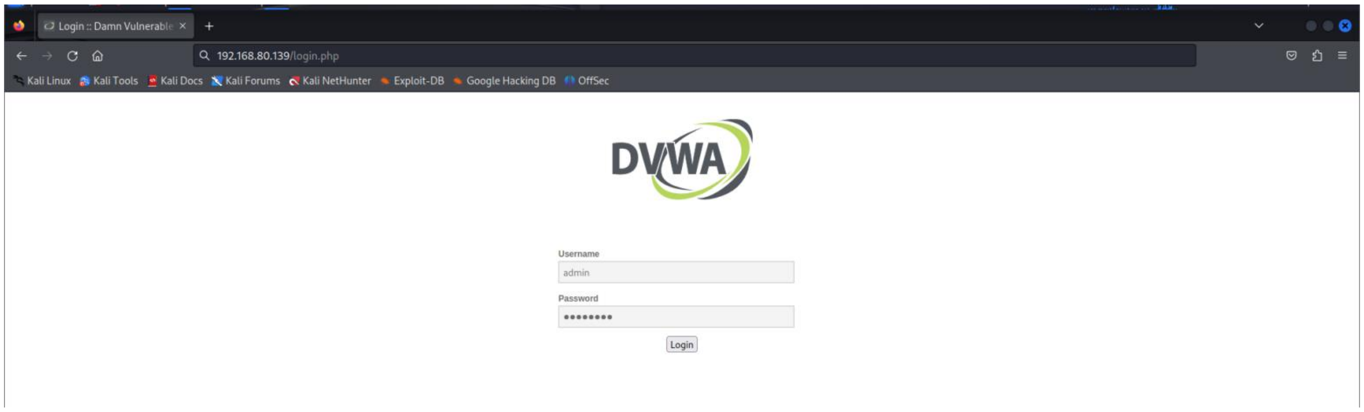
'guestbook' table was created.

Data inserted into 'guestbook' table.

Backup file /config/config.inc.php.bak automatically created

Setup successful!

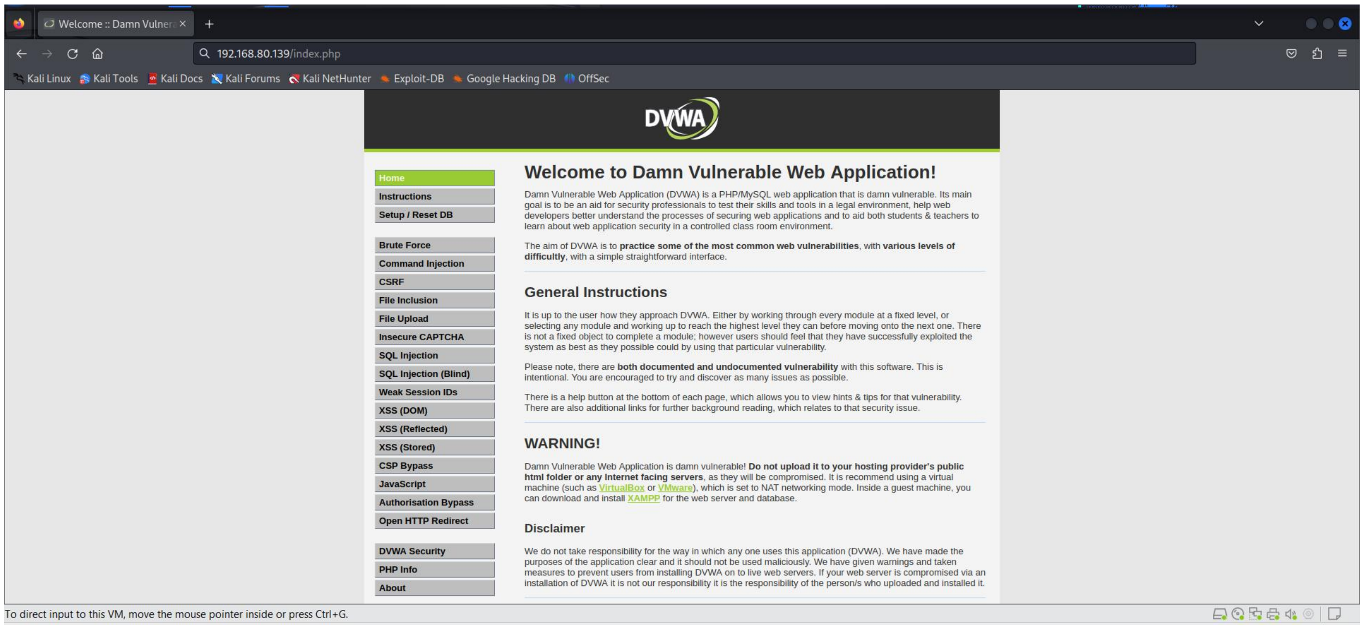
To direct input to this VM, move the mouse pointer inside or press Ctrl+G.


192.168.80.139/login.php

Username: admin

Password: ••••••••

Login


192.168.80.139/index.php

Welcome to Damn Vulnerable Web Application!

Damn Vulnerable Web Application (DVWA) is a PHP/MySQL web application that is damn vulnerable. Its main goal is to be an aid for security professionals to test their skills and tools in a legal environment, help web developers better understand the processes of securing web applications and to aid both students & teachers to learn about web application security in a controlled class room environment.

The aim of DVWA is to **practice some of the most common web vulnerabilities**, with **various levels of difficultly**, with a simple straightforward interface.

General Instructions

It is up to the user how they approach DVWA. Either by working through every module at a fixed level, or selecting any module and working up to reach the highest level they can before moving onto the next one. There is not a fixed object to complete a module; however users should feel that they have successfully exploited the system as best as they possible could by using that particular vulnerability.

Please note, there are **both documented and undocumented vulnerability** with this software. This is intentional. You are encouraged to try and discover as many issues as possible.

There is a help button at the bottom of each page, which allows you to view hints & tips for that vulnerability. There are also additional links for further background reading, which relates to that security issue.

WARNING!

Damn Vulnerable Web Application is damn vulnerable! **Do not upload it to your hosting provider's public html folder or any Internet facing servers**, as they will be compromised. It is recommend using a virtual machine (such as VirtualBox or VMware), which is set to NAT networking mode. Inside a guest machine, you can download and install XAMPP for the web server and database.

Disclaimer

We do not take responsibility for the way in which any one uses this application (DVWA). We have made the purposes of the application clear and it should not be used maliciously. We have given warnings and taken measures to prevent users from installing DVWA on to live web servers. If your web server is compromised via an installation of DVWA it is not our responsibility it is the responsibility of the person/s who uploaded and installed it.

To direct input to this VM, move the mouse pointer inside or press Ctrl+G.

3.2. Conclusion

- That is how we install DVWA using ansible!!! Ansible is a tool that helps prepare in advance and manage configurations. It is the simplest and easiest way to get started because it only uses SSH to connect to the Server and run pre-configured Tasks. Ansible also helps us easily convert a Bash script into tasks in Ansible for management. In addition, before running Tasks, users can preview the context and handle their limitations.

- In our project, however, there are some limitations, which is that we cannot automate changing the password of the MySQL root account. That's why we have to do it manually.

IV. Appendix

4.1. Task Assignment

Member	Task	Percent(%)
Phùng Đức Lương	Tìm tài liệu, cài đặt, cấu hình, demo, thuyết trình, làm slide.	100%
Ngô Minh Quân	Tìm tài liệu, cấu hình, demo, thuyết trình, làm slide.	100%
Chu Nguyễn Hoàng Phương	Tìm tài liệu, cài đặt, cấu hình, demo, thuyết trình.	100%

4.2. Self - Assessment

	1	2	3	4	Overall
Present				x	4
Theory				x	4
Report				x	4
Demo			x		3

4.3 Question & Answer

- Managed nodes có cần cài thêm gói/phần mềm gì không?

*-> **The answer is:*** Ansible does not require the installation of any additional software or packages on Managed nodes. Ansible uses SSH (or WinRM on Windows) connections to communicate and execute modules across Managed nodes. This makes Ansible simple and does not require installing agents or additional software on Managed nodes

- Làm thế nào để xử lý lỗi và giám sát trong Ansible ?

-> *The answer is:* Blocks and rescue work together to provide error-handling capabilities in Ansible. Use the rescue keyword in association with a block to define a set of tasks that will be executed if an error occurs in the block. We can use the rescue tasks to handle errors, log messages, or take other actions to recover from the error.

Here is an example:

```
---
- hosts: <hosts>
  tasks:
    - block:
        - <task1>
        - <task2>
        - <task3>
      rescue:
        - <rescue_task1>
        - <rescue_task2>
        - <rescue_task3>
      always:
        - <always_task>
```

We define tasks under the **block** keyword, which could be as simple as invoking the ansible.builtin.ping module, or we could have a combination of multiple tasks and including/importing roles.

The associated **rescue** keyword is where the playbook execution will be sent, for each host, if anything fails along the block.

Finally, the **always** section executes for all nodes, no matter if they succeed or fail.

- **About the monitoring in Ansible**, we can use Ansible Tower or third-party monitoring tools to automate Ansible monitoring. For example, Prometheus and Grafana can be used to monitor Ansible health and other metrics.