Ansible Project: Install LAMB stacks.

Ansible Project: Install LAMB stack using Ansible playbook:

Project description:

-A job ticket was submitted to me by a colleague, and it stated that the LAMB stack needed to be installed in 3 development servers. The ticket didn't specify a type of method to use to solve the problem, so this was a great opportunity to use Ansible to automate the task of installing the LAMB stack to the required servers.

-I am going to use a playbook to accomplish the required steps to solve the problem.

The playbook functions:

-Install all necessary packages like Apache(httpd), mariadb and php.

-Install a firewall and enable HTTP services.

-Start the Apache HTTPD web server.

-Start the MariaDB server.

-Download a Sample PHP page from a remote URL for testing purposes.

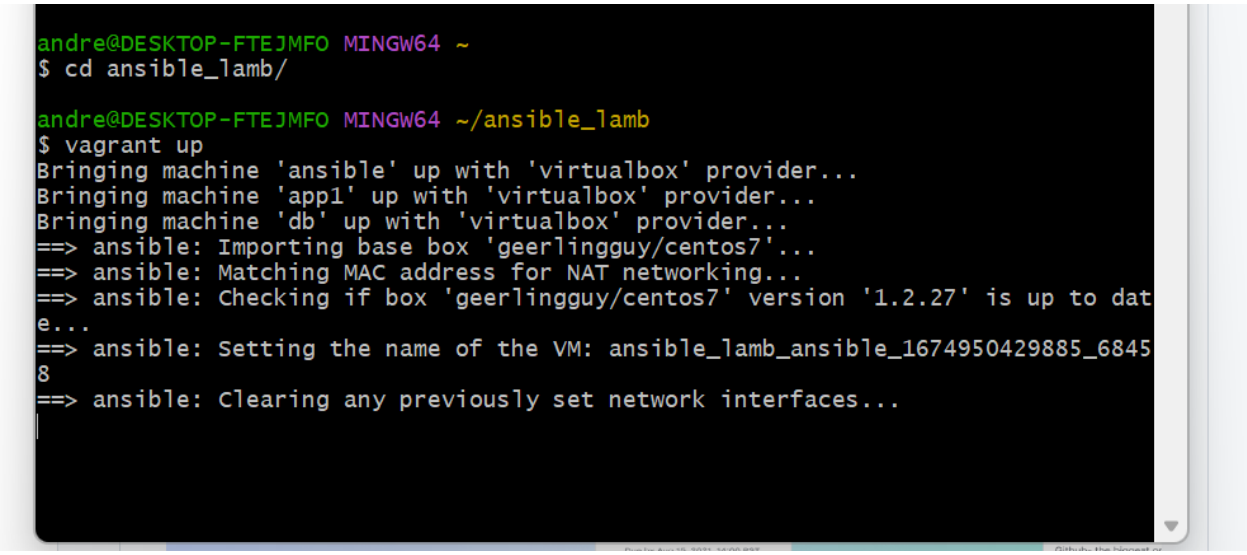
-Access the website we have built by accessing the URL.

Setting up the server:

-One server is going to be the master server which has Ansible running on it.

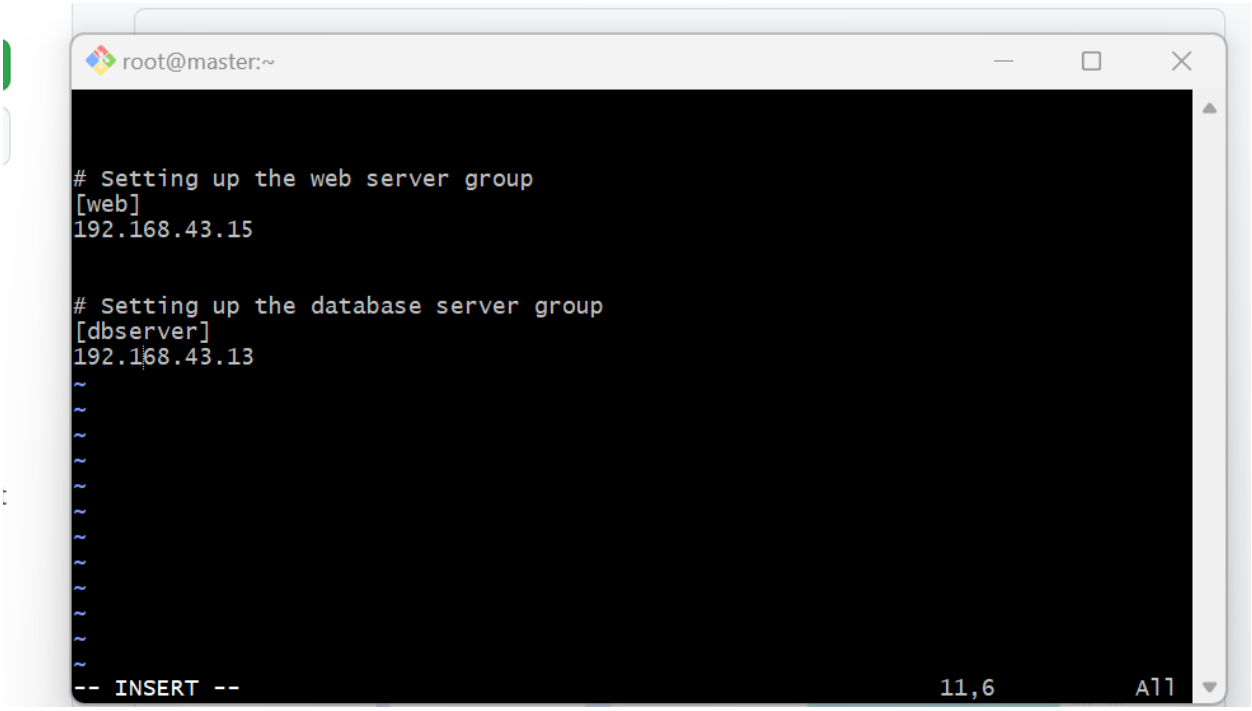
-The other two servers are going to be the nodes and use the master server to connect to both node servers

Creating 3 centos7 servers for the project: One master server, one webserver node server, and one DB node server.



-Instead of installing Ansible after the creation of the server. I decided to automate the installation task of Ansible by creating a custom Vagrantfile that would install Ansible into the master node whenever the server is created.

Setting the target machines on the host's config file in the master server



-Before I make any changes to the target machines, I need to make sure I can connect to them. I usually run a ping command to achieve the goal:

Command example:

cd /etc/ansible

ansible server name -m ping : ansible web01 -m ping

-If the connection was a success, we should see a message confirming it, but sometime you might encounter a permission denied or unreachable error. It just means that the master server does not have the necessary information to connect to the node servers. In this case, I just need to create a key in the master server and copy the public key to the target server:

-Use this command to generate the key:

-sudo ssh-keygen

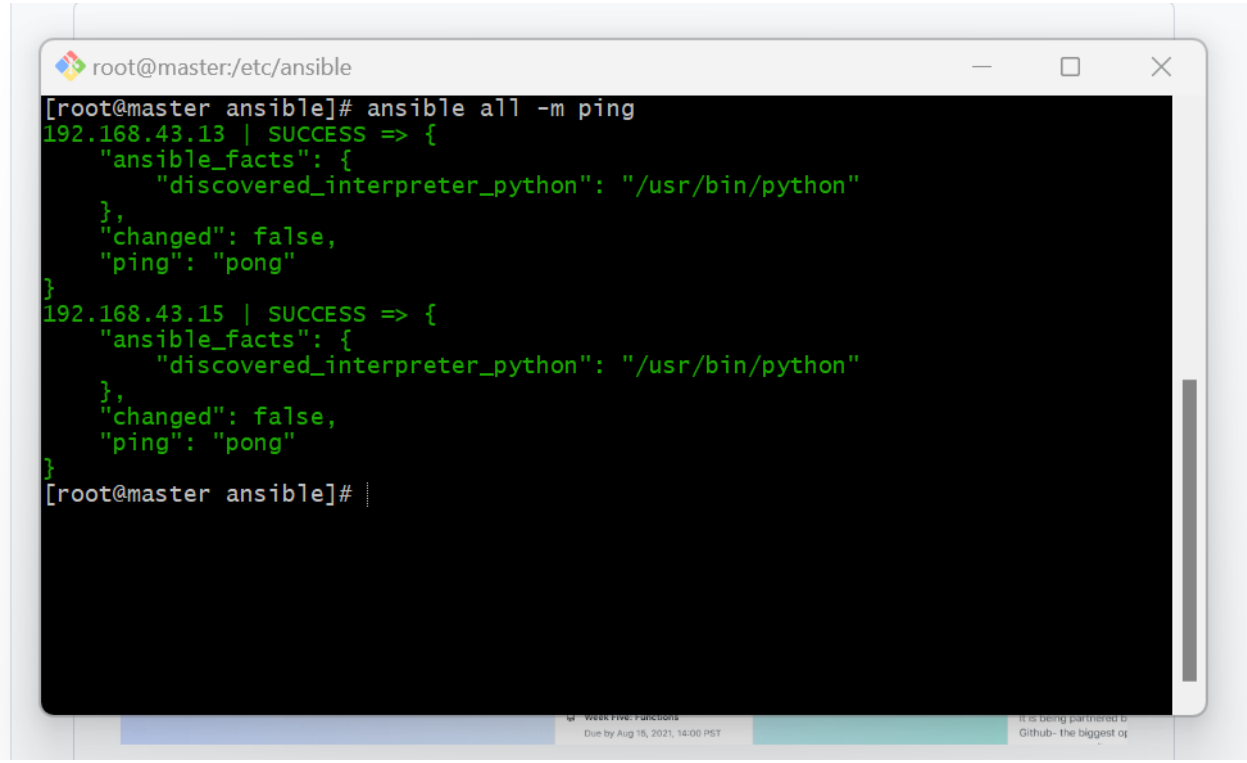
-Copy the keys to the target server:

-sudo ssh-copy-id serverIPaddress

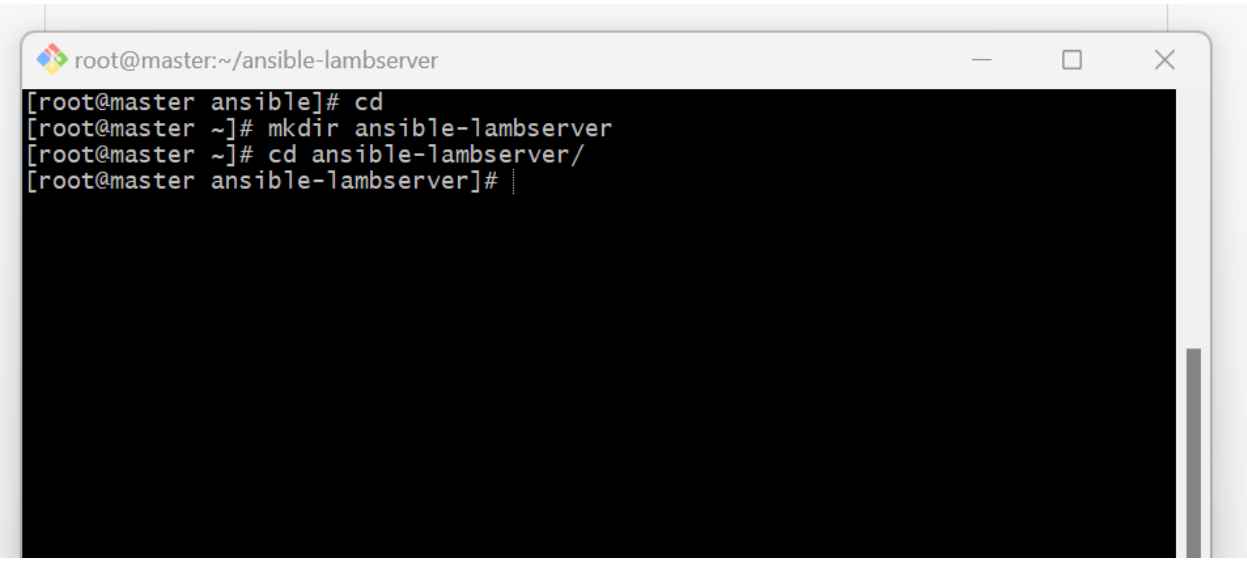
-sudo ssh-copy-id 192.168.42.10

-The above steps should fix the permission denied error and try to ping all the target servers again.

Pinging all the servers in the hosts/inventory file to make sure I can be able to make changes to the host servers



Let's create a directory in the home directory on my master server. I am going to do most of my work in this directory.



Now, I am about to create an inventory file of the target servers.

```
root@master:~/ansible-lambserver
[root@master ansible-lambserver]# ls
inventory.txt
[root@master ansible-lambserver]# cat inventory.txt


[lambservers]

192.168.43.13

192.168.43.15


[root@master ansible-lambserver]#
```

Because I decided to configure this process manually. I am going to create a file called ansible.cfg in the directory that I created earlier.

```
root@master:~/ansible-lambserver
[root@master ansible-lambserver]# ls
ansible.cfg  inventory.txt
[root@master ansible-lambserver]# cat ansible.cfg

[defaults]
host_key_checking = False
inventory=inventory.txt
interpreter_python=auto_silent
localhost_warning=false

[root@master ansible-lambserver]#
```

Validate and check the inventory file

```
root@master:~/ansible-lambserver
[root@master ansible-lambserver]# ansible-inventory --graph
@all:
  |--@lambservers:
  |  |--192.168.43.13
  |  |--192.168.43.15
  |--@ungrouped:
[root@master ansible-lambserver]#
```

The ansible.cfg file is working properly.

Check again if I can be able to ping the target servers

```
root@master:~/ansible-lambserver
[root@master ansible-lambserver]# ansible all -m ping
192.168.43.15 | SUCCESS => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python"
    },
    "changed": false,
    "ping": "pong"
}
192.168.43.13 | SUCCESS => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python"
    },
    "changed": false,
    "ping": "pong"
}
[root@master ansible-lambserver]#
```

Now, it's time to write the playbook to install the LAMB stack in all the target servers.

Command:

-$ touch lambstack.yml or

-$ vim lambstack.yml

After creating the playbook, I need to validate the syntax.

```
root@master:~/ansible-lambserver
[root@master ansible-lambserver]# ls
ansible.cfg  inventory.txt  lambstack.yml
[root@master ansible-lambserver]# ansible-playbook lambstack.yml --syntax-check

playbook: lambstack.yml
[root@master ansible-lambserver]#
```

Now, let me run the playbook

```
root@master:~/ansible-lambserver
ok: [192.168.43.15]

TASK [httpd enabled and running] ***********************************************
ok: [192.168.43.13]
ok: [192.168.43.15]

TASK [mariadb enabled and running] *********************************************
ok: [192.168.43.13]
ok: [192.168.43.15]

TASK [copy the php page from remote using get_url] *****************************
changed: [192.168.43.13]
changed: [192.168.43.15]

TASK [test the webpage/website we have setup] **********************************
ok: [192.168.43.15]
ok: [192.168.43.13]

PLAY RECAP *********************************************************************
192.168.43.13              : ok=9    changed=1    unreachable=0    failed=0    s
192.168.43.15              : ok=9    changed=1    unreachable=0    failed=0    s

[root@master ansible-lambserver]#
```

-The playbook ran successfully

Now, let's check on the host servers.

Not secure | 192.168.43.13/index.php

The Best PHP Examples

Not secure | 192.168.43.15/index.php

The Best PHP Examples

LAMB stack is installed on all the host servers.