

Ansible Project: Reboot Servers

Ansible Project:

Rebooting servers using Ansible.

Problem Statement:

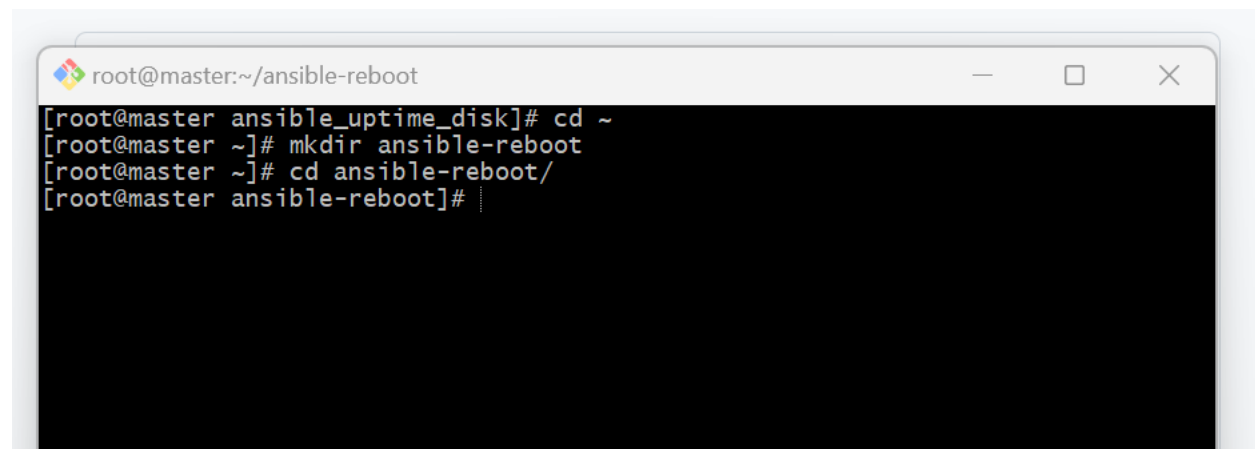
-Some companies reboot all their servers once every week for good maintenance of the server. This process can be tedious doing it manually especially if you are working with a lot of servers. A company submitted a job ticket for automating the task of rebooting multiple servers using Ansible.

Solution Statement:

-For demonstration purposes, I am going to use three servers to solve the problem. The concept is the same if you are working with 3 servers or 100 servers. One of my servers will be used as the master node and the other two as the target nodes. I am going to create an Ansible playbook that will reboot the target servers from the inventory or host file.

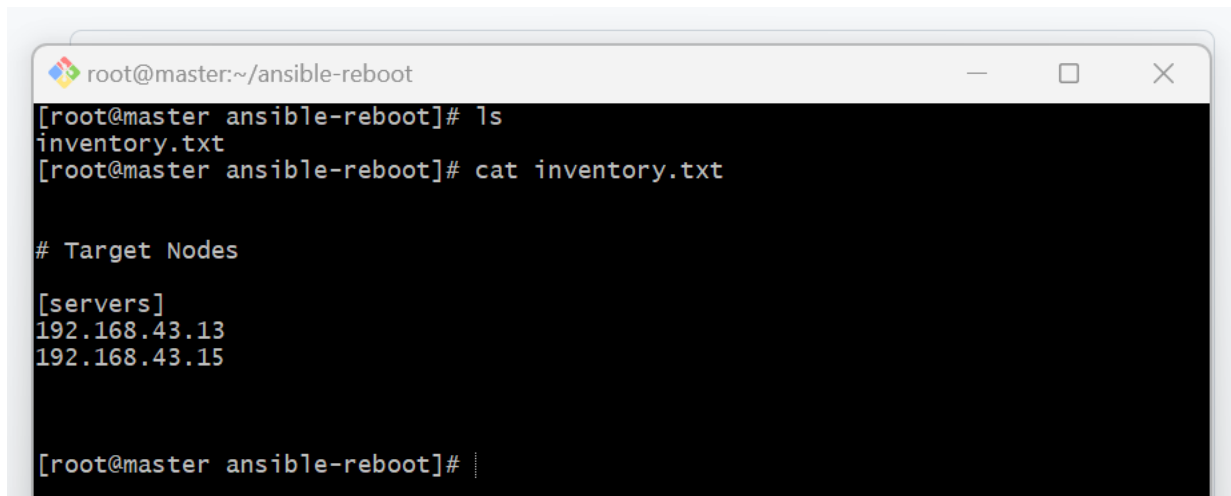
Some of the reasons why a company would reboot their servers every week are regular restart of a server makes the environment more stable and better performing. Also, regular reboots protect the business from outages and downtime.

Creating a directory for Ansible files including the playbook

A terminal window with a title bar showing 'root@master:~/ansible-reboot'. The terminal text shows the following commands and their outputs: '[root@master ansible_uptime_disk]# cd ~', '[root@master ~]# mkdir ansible-reboot', '[root@master ~]# cd ansible-reboot/', and '[root@master ansible-reboot]# |'.

```
root@master:~/ansible-reboot
[ root@master ansible_uptime_disk ]# cd ~
[ root@master ~ ]# mkdir ansible-reboot
[ root@master ~ ]# cd ansible-reboot/
[ root@master ansible-reboot ]# |
```

Creating an inventory file to keep track of all the hosts.

A terminal window titled 'root@master:~/ansible-reboot' with standard window controls. The user runs 'ls' and 'cat inventory.txt'. The output of 'cat' shows a comment '# Target Nodes', a group '[servers]', and two IP addresses: '192.168.43.13' and '192.168.43.15'.

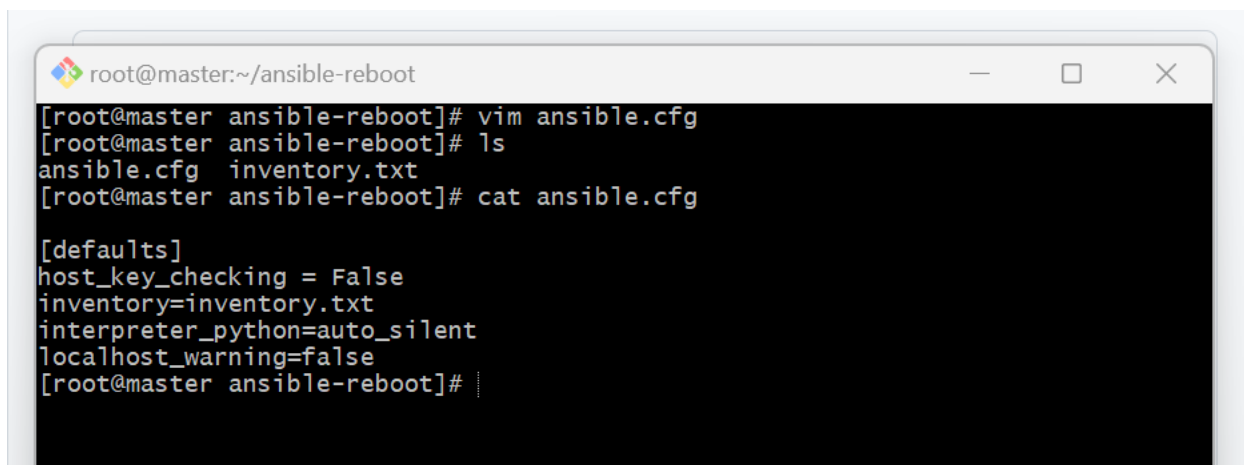
```
root@master:~/ansible-reboot
[root@master ansible-reboot]# ls
inventory.txt
[root@master ansible-reboot]# cat inventory.txt

# Target Nodes

[servers]
192.168.43.13
192.168.43.15

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

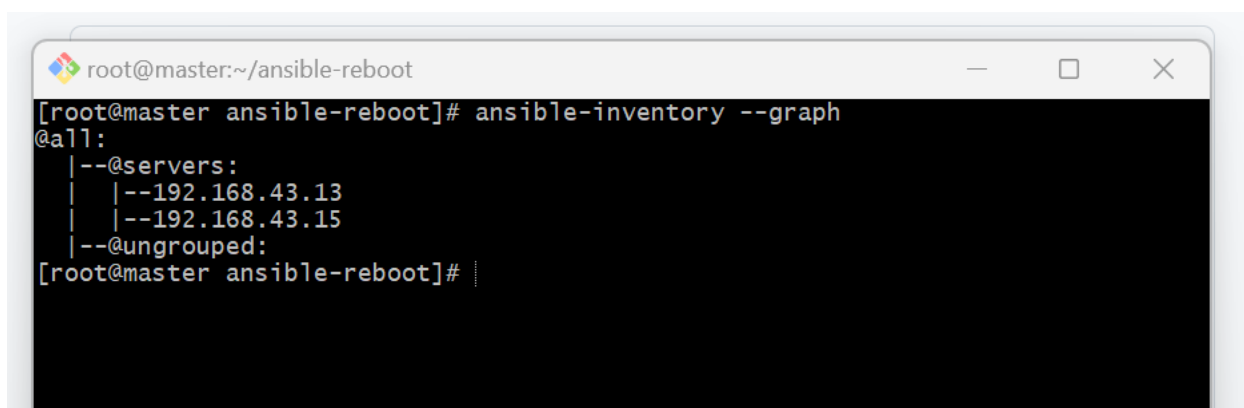
Creating a configuration for ansible

A terminal window titled 'root@master:~/ansible-reboot' with standard window controls. The user runs 'vim ansible.cfg', 'ls', and 'cat ansible.cfg'. The output of 'cat' shows the '[defaults]' section with settings: 'host_key_checking = False', 'inventory=inventory.txt', 'interpreter_python=auto_silent', and 'localhost_warning=false'.

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

[defaults]
host_key_checking = False
inventory=inventory.txt
interpreter_python=auto_silent
localhost_warning=false
[root@master ansible-reboot]#
```

Validate the inventory

A terminal window titled 'root@master:~/ansible-reboot' with standard window controls. The user runs 'ansible-inventory --graph'. The output shows a tree structure starting with '@all:', followed by '@servers:' containing the two IP addresses, and '@ungrouped:'.

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

Checking the connectivity of all the target nodes

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

Checking the playbook syntax before running it

```
root@master:~/ansible-reboot
[root@master ansible-reboot]# ansible-playbook reboot.yml --syntax-check
playbook: reboot.yml
[root@master ansible-reboot]#
```

Playbook output

```
root@master:~/ansible-reboot

PLAY [servers] *****

TASK [Gathering Facts] *****
ok: [192.168.43.13]
ok: [192.168.43.15]

TASK [restart server] *****
changed: [192.168.43.13]
changed: [192.168.43.15]

TASK [waiting for the server to come back] *****
ok: [192.168.43.15 -> localhost]
ok: [192.168.43.13 -> localhost]

PLAY RECAP *****
192.168.43.13      : ok=3    changed=1    unreachable=0    failed=0    s
192.168.43.15      : ok=3    changed=1    unreachable=0    failed=0    s
```

Now, we can be able to reboot the target server automatically.