Ansible Tutorial Report

Contents

Introduction 3
Objectives 3
Setup and Configuration 3
Task 3
Execution of Playbooks 4
Results and Discussion 4
Conclusion 4
Screenshots: 5

Introduction

Within this method, the key facets of the automation tool namely ‘Ansible’ has been described in the following sections of the report. Newer operating systems like RHEL7 have inbuilt features like kickstart that help in tasks like backups, monitoring network performance, and even deploying updates across the networks, which Ansible makes much easier.

Objectives

The main aim of the exercise was to check and measure network performance using some Ansible playbooks on availability of local Ubuntu system based on VirtualBox where the focus was made on checking the CPU load and network interfaces.

Setup and Configuration

System Setup: Ubuntu 20. 04 on a VirtualBox hosted on a Windows host machine/paragraph begins/To achieve this the author installed Ubuntu 04 on a VirtualBox that runs on a Windows PC.

Ansible Installation: Ansible was installed from the official and recognized PPA, along with the system that was configured with essential deploys.

Task

Network Management Task: Monitoring Network Performance (Implement to PC computer networking)

- **Playbook Name**: **local_monitor.yml**
- **Tasks Implemented**:

- Fetch and display CPU load using **/proc/loadavg**.
- Monitor and report memory usage with **free -m**.
- Retrieve network interface details using **ip a** command.

Execution of Playbooks

The playbook **local_monitor.yml** was executed using the command:

```
ansible-playbook local_monitor.yml -i hosts.ini
```

Results were captured and analyzed, highlighting the system's current operational status.

Results and Discussion

There were some issues when the playbook execution beginning with command not found: fixed with change ifconfig to ip a. It is mandatory to keep a record of every process of troubleshooting done and any form of changes made to the said playbook.

Conclusion

From the above analysis, it is clear that Ansible has great automation in the management and monitoring of configuration systems within a network. From the work accomplished, the versatility of the Ansible structure and the playbooks specifically became clear when performing dynamic and intricate network-related tasks.

Screenshots:

Open | local_monitor.yml ~/~ansible-projects/network_monitoring | Save

```yaml
- name: Monitor Local System
  hosts: localhost
  gather_facts: no
  tasks:
    - name: Get CPU load
      command: cat /proc/loadavg
      register: cpu_load

    - name: Display CPU load
      debug:
        msg: "CPU Load: {{ cpu_load.stdout }}"

    - name: Check memory usage
      command: free -m
      register: memory_usage

    - name: Display memory usage
      debug:
        msg: "Memory Usage: {{ memory_usage.stdout }}"

    - name: Fetch network interfaces
      command: ip a
      register: ifconfig_output

    - name: Display network interfaces
      debug:
        msg: "{{ ifconfig_output.stdout }}"
```

YAML | Tab Width: 8 | Ln 22, Col 20 | INS

Figure 01 : local_monitor.yml

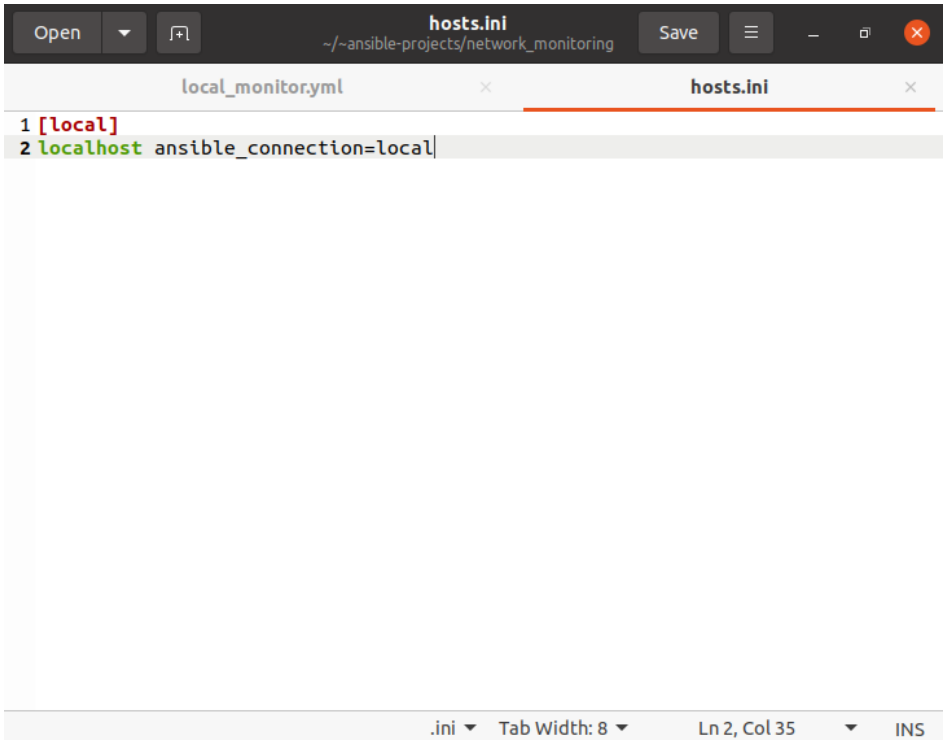

Figure 02: hosts.ini

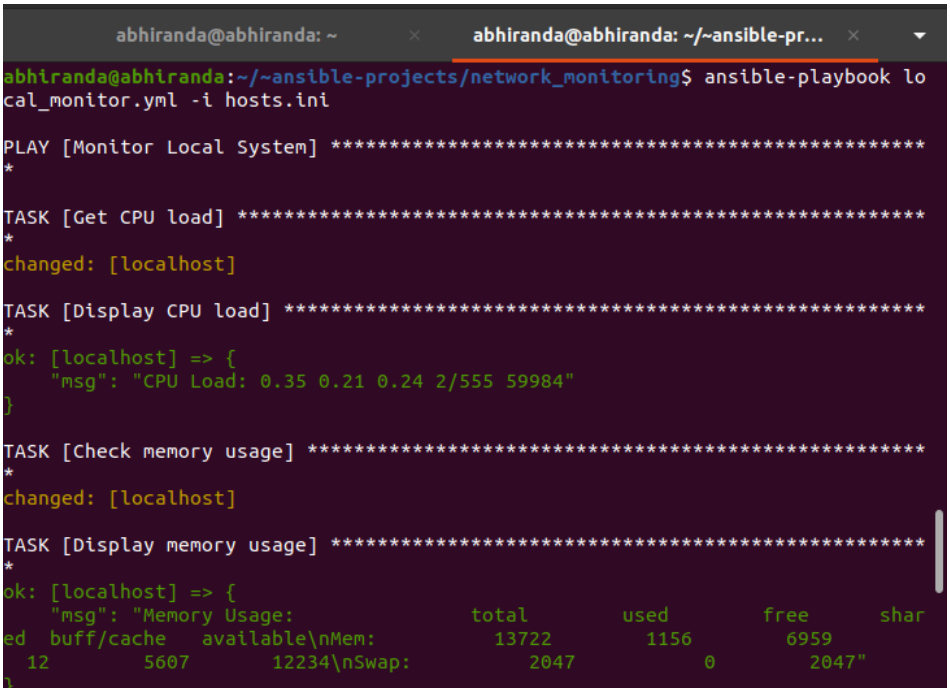

Figure 03: Output (1)

```
TASK [Display CPU load] ******************************************************
*
ok: [localhost] => {
    "msg": "CPU Load: 0.35 0.21 0.24 2/555 59984"
}

TASK [Check memory usage] ****************************************************
*
changed: [localhost]

TASK [Display memory usage] **************************************************
*
ok: [localhost] => {
    "msg": "Memory Usage:               total        used        free      shar
ed  buff/cache   available\nMem:          13722        1156        6959
  12        5607       12234\nSwap:          2047           0        2047"
}

TASK [Fetch network interfaces] **********************************************
*
changed: [localhost]

TASK [Display network interfaces] ********************************************
*
ok: [localhost] => {
    "msg": "1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN
```

Figure 03: Output (2)

```
TASK [Fetch network interfaces] **********************************************
*
changed: [localhost]

TASK [Display network interfaces] ********************************************
*
ok: [localhost] => {
    "msg": "1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN
 group default qlen 1000\n    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:0
0:00\n    inet 127.0.0.1/8 scope host lo\n       valid_lft forever preferred_lf
t forever\n    inet6 ::1/128 scope host \n       valid_lft forever preferred_lf
t forever\n2: enp0s3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel
 state UP group default qlen 1000\n    link/ether 08:00:27:aa:55:89 brd ff:ff:f
f:ff:ff:ff\n    inet 10.0.2.15/24 brd 10.0.2.255 scope global dynamic noprefixr
oute enp0s3\n       valid_lft 82278sec preferred_lft 82278sec\n    inet6 fe80::
a64c:5294:66a:f77f/64 scope link noprefixroute \n       valid_lft forever prefe
rred_lft forever"
}

PLAY RECAP *******************************************************************
*
localhost                  : ok=6    changed=3    unreachable=0    failed=0
skipped=0    rescued=0    ignored=0

abhiranda@abhiranda:~/~ansible-projects/network_monitoring$
```

Figure 03: Output (3)