Department of Computer Science and Engineering (Data Science)

AY: 2025-26

Semester: V

Subject: DevOps Laboratory (DJS23OLOE501)

Experiment 3a

(Configuration Management with Puppet and Ansible)

Name : Soham Kishor Walam **Roll No: D102**

Aim: To Perform Configuration Management using Ansible.

Theory:

Introduction to Configuration Management

Configuration Management (CM) is a critical practice in DevOps that automates the process of managing and maintaining software and infrastructure configurations across servers. It ensures that systems are deployed consistently, reducing human error, and allowing for version control, auditing, and repeatability.

Two popular CM tools in this domain are **Puppet** and **Ansible**. Both tools automate configuration tasks, but they differ in architecture and approach.

What is Ansible?

Ansible is a powerful, agentless automation tool that uses YAML-based playbooks to define system configurations. It uses SSH to connect to target machines, making it easy to use and secure.

Key Concepts:

- Inventory: A list of target hosts (e.g., IPs or hostnames).
- Playbook: A YAML file that defines tasks to run on hosts.
- Task: A single action (e.g., install a package).
- Role: A structured way to group playbooks, variables, files, templates, and handlers for complex setups.

Ansible Architecture

- Control node: Commands and Playbooks can run by invoking /usr/bin/ansible or /usr/bin/ansible-playbook, from any control node. You can use any computer that has Python installed on it as a control node. However, one cannot use a computer with Windows OS as a control node. One can have multiple control nodes.
- Managed nodes: Also, sometimes called "hosts", Managed nodes are the network devices (and/or servers) you manage with Ansible.
- Inventory: Also, sometimes called "hostfile", Inventory is the list of Managed nodes use to organize them. It is also used for creating and nesting groups for easier scaling.

Department of Computer Science and Engineering (Data Science)

Modules: These are the units of code executed by Ansible. Each module can be used for a specific purpose. One can invoke a single module with a task, or invoke several different modules in a playbook.

Tasks: The units of action in Ansible. One can execute a single task once with an ad-hoc command.

1. **Install Ansible on a Linux Server**

Step 1 – Create 2 VM Instances in GCP

We will use one as **control node** (where Ansible is installed) and the other as a **managed node**.

1. Go to **Google Cloud Console → Compute Engine → VM instances → Create Instance**.
2. Create **VM 1**:
    - Name: control-node
    - OS: Ubuntu 22.04 LTS
    - Machine type: e2-micro
    - Allow **HTTP/HTTPS** in firewall.
3. Create **VM 2**:
    - Name: managed-node
    - OS: Ubuntu 22.04 LTS
    - Machine type: e2-micro
    - Allow **HTTP/HTTPS** in firewall.

Step 2 – Install Ansible on Control Node

SSH into control-node:

sudo apt update

sudo apt install -y ansible

Verify:

ansible –version

Department of Computer Science and Engineering (Data Science)

Step 3 – Set Up SSH Access to Managed Node

On control-node:

ssh-keygen -t rsa -b 2048
(Press Enter for all prompts to accept defaults.)

Copy the public key to managed node:

Option 1: ssh-copy-id moreshraddha30@34.10.9.52

Option 2: Use GCP Console to Add Your Public Key

1. On your control-node, display your public key:
2. cat ~/.ssh/id_rsa.pub
3. Copy the entire line starting with ssh-rsa ....
4. In GCP Console, go to VM Instances → managed-node → Edit.
5. Scroll to SSH Keys, click Show and edit.
6. Paste your public key there, then click Save.

Copy key:

ssh-rsa
AAAAB3NzaC1yc2EAAAADAQABAAABAQCtCMZDZkfnjKYqdtJimOYsl5pwddsm6gk
ULozQ4WpPhevlt8OMZziVMqM0R5+cHFz2k87a7gRNfoYgosq6KcZmyLZccmSjytr9ahfu
DU0dnxeNA5WhhtvWDUdo+4hZpVxredDMgREzBGxBfG4Zi62Q59nclQm9ykPzr1DXV
UNCkcVJ/JqcJwbBHalZNiStlx2twtdHjwYoheZDLONVh5au8sxU1JtHWN54vTiUu+IJKc4
xbNLzMCfyBMWPZoWXqXG6NCVYXZV2OIGxkaVdAr4pABvrNWL6+mJSomRb0f5IF
bISQe1eOJM5YkiWcRJe3l0mX7JeC7tPnEfqJV8wyK0f moreshraddha30@control-node

Now you should be able to connect:

ssh moreshraddha30@34.10.9.52

```
moreshraddha30@control-node:~$ ssh moreshraddha30@34.10.9.52
The authenticity of host '34.10.9.52 (34.10.9.52)' can't be established.
ED25519 key fingerprint is SHA256:NY1D04WknspWIo+4DdZn2n7hPeP8vGpSX7nFDMBuDwE.
This key is not known by any other names
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '34.10.9.52' (ED25519) to the list of known hosts.
Welcome to Ubuntu 22.04.5 LTS (GNU/Linux 6.8.0-1034-gcp x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/pro

This system has been minimized by removing packages and content that are
not required on a system that users do not log into.

To restore this content, you can run the 'unminimize' command.

Expanded Security Maintenance for Applications is not enabled.

0 updates can be applied immediately.

Enable ESM Apps to receive additional future security updates.
See https://ubuntu.com/esm or run: sudo pro status


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

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

moreshraddha30@managed-node:~$ ^C
```

That means your **control-node successfully connected to the managed-node via SSH →** which is the exact requirement for Ansible setup

Now your setup looks like this:

- **control-node** (has Ansible installed) → connects to
- **managed-node** (target machine) → via SSH key

Exit managed-node

moreshraddha30@managed-node:~$ exit

logout

Connection to 34.10.9.52 closed.

On control node

Install nano:

sudo apt update

sudo apt install -y nano

nano inventory

[servers]

managed-node ansible_host=<managed-node-external-ip> ansible_user=moreshraddha30

[servers]

managed-node ansible_host=34.10.9.52 ansible_user=moreshraddha30

Save and exit (CTRL+O, Enter, CTRL+X).

Test Ansible connectivity

ansible -i inventory servers -m ping

```
moreshraddha30@control-node:~$ ansible -i inventory servers -m ping
managed-node | SUCCESS => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python3"
    },
    "changed": false,
    "ping": "pong"
}
```

That output means:

- Your control-node can now talk to your managed-node over SSH without a password.
- The inventory file is set up correctly.

2. Write Ansible Playbook to Create Users & Install Software

mkdir ~/ansible_project

cd ~/ansible_project

nano setup.yml

- name: Setup users and install software

  hosts: servers

  become: yes

  tasks:

    - name: Create a new user

      user:

```
    name: devuser
    state: present
    shell: /bin/bash

- name: Install basic software
  apt:
    name:
      - git
      - curl
      - htop
    state: present
    update_cache: yes
```

nano inventory

```
[servers]
managed-node ansible_host=34.10.9.52  ansible_user=moreshraddha30
```

Save and exit (CTRL+O, Enter, CTRL+X).

Run the playbook

ansible-playbook -i inventory setup.yml

```
moreshraddha30@control-node:~/ansible_project$ ansible-playbook -i inventory setup.yml

PLAY [Setup users and install software] ****************************************************************

TASK [Gathering Facts] *********************************************************************************
ok: [managed-node]

TASK [Create a new user] *******************************************************************************
changed: [managed-node]

TASK [Install basic software] **************************************************************************
changed: [managed-node]

PLAY RECAP *********************************************************************************************
managed-node               : ok=3    changed=2    unreachable=0    failed=0    skipped=0    rescued=0    ignored
=0

moreshraddha30@control-node:~/ansible_project$
```

Step 3 – Verify on managed-node

SSH into managed-node and check:

```
# Check if user was created
cat /etc/passwd | grep devuser
```

That confirms the Ansible playbook worked successfully — the user devuser was created on your managed node.

 # Check if packages are installed

git --version

curl --version

htop –version

```
moreshraddha30@managed-node:~$ cat /etc/passwd | grep devuser
devuser:x:1002:1003::/home/devuser:/bin/bash
moreshraddha30@managed-node:~$ ^C
moreshraddha30@managed-node:~$ git --version
git version 2.34.1
moreshraddha30@managed-node:~$ curl --version
curl 7.81.0 (x86_64-pc-linux-gnu) libcurl/7.81.0 OpenSSL/3.0.2 zlib/1.2.11 brotli/1.0.9 zstd/1.4.8 libidn2/2.3.2
 libpsl/0.21.0 (+libidn2/2.3.2) libssh/0.9.6/openssl/zlib nghttp2/1.43.0 librtmp/2.3 OpenLDAP/2.5.19
Release-Date: 2022-01-05
Protocols: dict file ftp ftps gopher gophers http https imap imaps ldap ldaps mqtt pop3 pop3s rtmp rtsp scp sftp
 smb smbs smtp smtps telnet tftp
Features: alt-svc AsynchDNS brotli GSS-API HSTS HTTP2 HTTPS-proxy IDN IPv6 Kerberos Largefile libz NTLM NTLM_WB
PSL SPNEGO SSL TLS-SRP UnixSockets zstd
moreshraddha30@managed-node:~$ htop --version
htop 3.0.5
moreshraddha30@managed-node:~$
```

3. Use Ansible Roles to Manage Complex Configurations
Step 1 – Create Role Structure

1) Create a project (if you don't have one)
   mkdir -p ~/ansible_project && cd ~/ansible_project
2) Create the two roles
   ansible-galaxy init user_setup
   ansible-galaxy init software_install
   This will create two directories like this:
   ~/ansible_project/
   ├── site.yml
   ├── user_setup/
   │ ├── tasks/
   │ │ └── main.yml
   │ └── ...
   └── software_install/
   ├── tasks/
   │ └── main.yml
   └── ...

Step 2 – Configure Roles

Edit **user_setup/tasks/main.yml**

For user_setup

nano user_setup/tasks/main.yml

```
- name: Create a new user
  user:
    name: devuser
    state: present
    shell: /bin/bash
```

Edit **software_install/tasks/main.yml**

For software_install

nano software_install/tasks/main.yml

```
- name: Install basic software
  apt:
    name:
      - git
      - curl
      - htop
    state: present
    update_cache: yes
```

Step 3 – Create Main Playbook Using Roles

nano site.yml

```
- name: Server configuration with roles
  hosts: servers
  become: yes
  roles:
    - user_setup
    - software_install
```

Step 4 – Run Roles

ansible-playbook -i inventory site.yml

After this, log in to managed-node and verify:

User exists:

cat /etc/passwd | grep devuser

Department of Computer Science and Engineering (Data Science)

Software installed:

git --version

curl --version

htop –version

Lab experiment to be performed in this session:

Configure and Deploy NGINX Server using Ansible Roles

Tasks to Perform:

1. Create two Ansible roles:
   - user_setup → Create a new user named webadmin.
   - nginx_setup → Install and configure the NGINX web server.
2. In the role nginx_setup, perform the following:
   - Install nginx.
   - Ensure the nginx service is enabled and running.
   - Deploy a simple index.html page with the message:
   - Welcome to NGINX configured by Ansible!
3. Create a main playbook site.yml that applies both roles (user_setup and nginx_setup) on the managed node(s).
4. Run the playbook using the inventory file.

Verification:

- Confirm that the user webadmin is created on the managed node.
- Confirm that NGINX service is running.
- Access the managed node's IP in a browser or use curl http://<managed-node-ip> and verify the custom web page content.

Output:

- Screenshots of role directory structure.
- Contents of site.yml.
- Screenshot of playbook execution.
- Screenshot of webpage output (via curl/browser).

Department of Computer Science and Engineering (Data Science)

```
 GNU nano 6.2                                    inventory
[servers]
managed-node ansible_host=34.30.68.44 ansible_user=walamsoham
```

SSH-in-browser

```
 GNU nano 6.2
- name: Configure server using roles
  hosts: servers
  become: yes
  roles:
    - user_setup
    - nginx_setup
```

```
 GNU nano 6.2                              user_setup/tasks/main.yml
- name: Create a new user
  user:
    name: webadmin
    state: present
    shell: /bin/bash
```

SSH-in-browser

```
 GNU nano 6.2                                        nginx_setup/tas
- name: Install nginx
  apt:
    name: nginx
    state: present
    update_cache: yes

- name: Ensure nginx is running and enabled
  service:
    name: nginx
    state: started
    enabled: yes

- name: Deploy custom index.html
  copy:
    dest: /var/www/html/index.html
    content: "Welcome to NGINX configured by Ansible!\n"
    owner: www-data
    group: www-data
    mode: '0644'
```

```
 GNU nano 6.2                              user_setup/tasks/main.yaml
- name : Create a new user
  user:
    name: webadmin
    state: present
    shell: /bin/bash
```

```
walamsoham@control-node:~$ ansible-galaxy init nginx_setup
- Role nginx_setup was created successfully
```

```
walamsoham@control-node:~$ ansible-galaxy init user_setup
- Role user_setup was created successfully
```

```
moksh jain_10a_m@control-node:~/ansible_project$ ansible-playbook -i inventory site.yml

PLAY [Configure server using roles] ********************************************************************

TASK [Gathering Facts] *********************************************************************************
ok: [managed-node]

TASK [user_setup : Create a new user] ******************************************************************
changed: [managed-node]

TASK [nginx_setup : Install nginx] *********************************************************************
changed: [managed-node]

TASK [nginx_setup : Ensure nginx is running and enabled] ***********************************************
ok: [managed-node]

TASK [nginx_setup : Deploy custom index.html] **********************************************************
changed: [managed-node]

PLAY RECAP *********************************************************************************************
managed-node               : ok=4    changed=3    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0

moksh jain_10a_m@control-node:~/ansible_project$
```