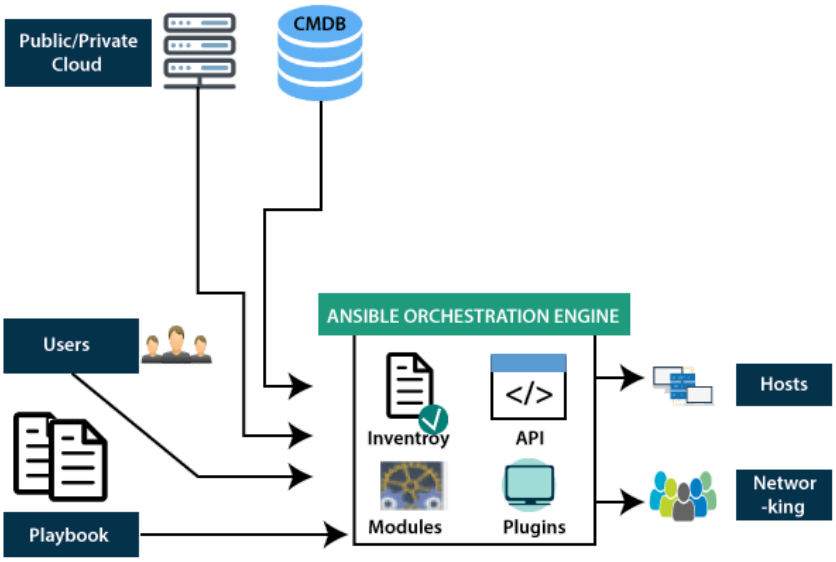
DEVOPS Training

ANSIBLE:

Ansible is an open-source automation tool that makes task orchestration, application deployment, and system management and configuration easier. It is intended to be simple to use, particularly for individuals who are new to DevOps.It is an tool that can significantly increase the consistency and efficiency of your IT settings, and it offers countless automation opportunities across hybrid clouds, on-premises infrastructure, and the Internet of Things.

Ansible Architecture:

Ansible is an automation tool with a straightforward architecture: managed nodes, which are the servers or computers that are being setup, and control nodes, where Ansible is deployed and runs instructions. These managed nodes are listed in the inventory file, and different Ansible settings are set in the configuration file (ansible.cfg). Roles group these tasks into reusable components, and playbooks, which are written in YAML, specify the tasks and configurations to be implemented. Variables provide dynamic configurations, handlers initiate operations only upon notification, and modules carry out specified tasks. Plugins expand on features, and templates generate dynamic files with the Jinja2 engine. Ansible's architecture guarantees that it is an effective and intuitive tool for automating IT processes.



Inventory-

The hosts or nodes you wish to control are listed in a file or dynamic source called the inventory in Ansible. In order to apply configurations or tasks selectively, it can be grouped into categories (such as web servers, databases) and contains information such as IP addresses and hostnames. This inventory is essential to effectively manage infrastructure since it tells Ansible which computers to target for automation activities.

Modules-

Ansible dispersed the Ansible module scripts and connected the nodes. Ansible runs the modules and removes them when it's done. These modules don't need a database or servers to run on any kind of computer. To monitor content changes, you can use a version control system, a terminal, or the text editor of your choice.

Playbooks-

Playbooks are composed of your written code that specifies the tasks and is executed by Ansible. They are written in YAML format. Playbooks also allow you to start the tasks both synchronously and asynchronously.

Installing ansible in ubuntu:

```
$ sudo apt update
$ sudo apt install software-properties-common
$ sudo add-apt-repository --yes --update ppa:ansible/ansible
$ sudo apt install ansible
```

Inventory file hosts:

Add IP address of node to the ansible server hosts in directory /etc/ansible/hosts

Ansible vault:

With the help of Ansible Vault, you can encrypt private information like passwords inside of your playbooks, roles, and variables to make sure it stays safe when being utilized for automated activities. Strong encryption is provided to shield data from unwanted access. Data can be encrypted individually files or specific variables, providing for flexibility. It also easily integrates with playbooks by decrypting data automatically at runtime. A password or key is needed to access encrypted data and is necessary for decryption.

Commands:

ansible-vault encrypt secrets.yml

This will prompt you to enter a password, which will be used to encrypt the file

ansible-vault decrypt secrets.yml

This will prompt you to enter a password, which will be used to decrypt the file

ansible-vault edit secrets.yml

This will prompt you to enter a password, which will be used to edit the file

Running Playbooks with Encrypted Data:

When running playbooks that include encrypted files, you must provide the vault password using the --ask-vault-pass flag, or specify a password file with the --vault-password-file flag:

ansible-playbook playbook.yml --ask-vault-pass

If using a password file:

ansible-playbook playbook.yml --vault-password-file vault_pass.txt

Ansible roles:

Playbooks can be more manageable and reusable when automation code is arranged and reused in a modular manner using Ansible roles. A set of tasks, handlers, variables, templates, and other files arranged into a certain directory structure make up each role. Roles assist you in decomposing intricate playbooks into more manageable, reusable parts.

Creating a role:

ansible-galaxy init my_role

Removing a role:

ansible-galaxy remove <role_name>

List installed roles:

ansible-galaxy list

To find the roles in directory we have to use /etc/ansible/roles/httpd_setup

A typical Ansible role has the following directory structure:

```
roles/
  my_role/
    defaults/
      main.yml
    files/
      myfile.conf
    handlers/
      main.yml
    meta/
      main.yml
    tasks/
      main.yml
    templates/
      mytemplate.j2
    vars/
      Main.yml
```

Making os ssh connection between server and node:

Installing SSH server:

sudo apt update

sudo apt install openssh-server

sudo systemctl status ssh

sudo systemctl start ssh

Configuring ssh server going to directory sudo nano /etc/ssh/sshd_config

Port: To change the default SSH port (22), modify the Port directive.

PermitRootLogin: To control root login, adjust the PermitRootLogin directive (e.g., no to disable root login).

PasswordAuthentication**:** To enable or disable password-based login, modify PasswordAuthentication (set to yes to enable, no to disable).

sudo systemctl restart ssh

Configure SSH Key-Based Authentication:

ssh-keygen -t rsa -b 2048

This command creates a public key (`id_rsa.pub`) and a private key (`id_rsa`) in the `~/.ssh/` directory by default.

sudo systemctl status sshd

sudo systemctl start sshd

ssh-copy-id user@managed-node

Alternatively, manually append your public key (`~/.ssh/id_rsa.pub`) to the `~/.ssh/authorized_keys` file on the managed node:

cat ~/.ssh/id_rsa.pub | ssh user@managed-node 'cat >> ~/.ssh/authorized_keys'

Connect to another machine by command:

ssh user@managed-node-ip

Use Ansible commands to test connectivity or run playbooks:

ansible all -m ping -i inventory

This command will test the SSH connectivity and Ansible configuration by pinging the managed node.

Note:-

Check Permissions: Ensure `~/.ssh` directory has `700` permissions and `~/.ssh/authorized_keys` file has `600` permissions.

Check SSH Configuration: Ensure the SSH server on the managed node is configured to accept key-based authentication (`/etc/ssh/sshd_config`).

Simple Ansible playbook to automate copying a built Java JAR or Python application from the server machine to the node machine and then running it:

```
---
- name: Copying files from one server to another
 hosts: all

 tasks:
 - name: Copy files
   copy:
     src: /home/priyam/Desktop/playbooks/hello1.py
     dest: /home/priyam/Desktop/ansible_files_from_ubuntu/
     mode: '664'

 - name: Running the python file of priyam
   command: python3
/home/priyam/Desktop/ansible_files_from_ubuntu/hello.py
   register: priyam_out

 - name: Displaying output
   debug:
     var: priyam_out.stdout_lines
```

Using Ansible vault for encrypting playbook.yml:

After giving command ansible-vault encrypt passoword.yml going to the current directory

```
$ANSIBLE_VAULT;1.1;AES256
30333663383337363435316532363830386337623133636563363638393266366136313737
376563
30313537356662396434643835353161663436636235613 00a3564616133316636306231 32
353834
37653063306665376633626131333064646262336564386137346365373534356639343761
633466
31653666393765383 40a3736613765343866643665313436653064333335326637613931 64
633537
37336233353730396638386532306539353638363766346231356438653434303762
```

Ansible var:
Variables, also known as vars, are used in Ansible to manage and store dynamic data that may be referred to in roles, tasks, and playbooks, increasing the flexibility and reusability of automation. Aside from being defined directly in playbooks under the vars section, variables can also be created within roles in files like defaults/main.yml or vars/main.yml, in independent YAML files included via vars_files, or in inventory files to apply settings to particular hosts or groups. Furthermore, the -e argument allows variables to be provided straight through the command line. The Jinja2 syntax ({{ variable_name }}) can be used to access them, and they facilitate dynamic deployment and configuration.
Example:

```
- hosts: all
  vars:
    app_name: "my_app"
    app_version: "1.0.0"
  tasks:
    - name: Create a directory for the application
      file:
        path: "/opt/{{ app_name }}"
        state: directory

    - name: Deploy application version
      copy:
        src: "{{ app_name }}-{{ app_version }}.jar"
        dest: "/opt/{{ app_name }}/"
```

Copying files to AWS s3 bucket from server to Aws:
Installing aws cli in ubuntu- sudo apt install awscli
Checking version-
aws --version
Configure Credentials of AWS CLI:
aws configure
This will create a configuration file in `~/.aws/config` and credentials file in `~/.aws/credentials`.

This will create a configuration file in `~/.aws/config` and credentials file in `~/.aws/credentials`.
This command syncs the contents of the local directory with the S3 bucket, only uploading new or modified files.

Copying files from local machine to aws S3 bucket:

```
---
- name: Copy JAR file to S3
 hosts: localhost

 vars:
   s3_bucket: "priyam-bucket-s3"

 tasks:
   - name: Copy JAR file to S3
     command: aws s3 cp /home/priyam/Desktop/playbooks/my-app.jar s3://{{
s3_bucket }}/my-app.jar
```

Jenkins pipeline for above task:
First we have to install the ssh and ansible plugin in Jenkins server machine.
Playbook:

```
---
- name: Copying files from one server to another
  hosts: all

  tasks:
  - name: Copy files
    copy:
      src: /var/lib/jenkins/workspace/CALSOFT-Ansible_assignment3/hello1.py
      dest: /var/lib/jenkins/
      mode: '777'

  - name: Running the python file of priyam
```

```
    command: python3 /var/lib/jenkins/hello1.py
    register: priyam_out

  - name: Displaying output
    debug:
      var: priyam_out.stdout_lines
```

Pipeline in Jenkins server machine:

```
pipeline {
    agent any

    environment {
        ANSIBLE_INVENTORY =
'/var/lib/jenkins/workspace/CALSOFT-Ansible_assignment3/hosts'
        ANSIBLE_PLAYBOOK =
'/var/lib/jenkins/workspace/CALSOFT-Ansible_assignment3/Ansible_assignment1.yml'
        //ANSIBLE_SSH_ARGS = '-o StrictHostKeyChecking=no' // Disable host key checking
    }

    stages {

        stage('Run Ansible Playbook') {
            steps {
                script {

                    sh "ansible-playbook Ansible_assignment1.yml"
                    sh 'pwd'
                    //sh "ansible-playbook -i ${ANSIBLE_INVENTORY} ${ANSIBLE_PLAYBOOK}
--ssh-common-args '${ANSIBLE_SSH_ARGS}'"
                }
            }
        }
    }

    post {
```

```
        success {
            echo 'Playbook executed successfully!'
        }
        failure {
            echo 'Playbook execution failed.'
        }
    }
}
```

Installing application ex-Nginx through ansible playbook from server to node machine:

```
---
- name: First yml playbook of mine
 hosts: all

 tasks:
 - name: Installing nginx
   apt:
     name: nginx
     state: present

 - name: Start the nginx service
   service:
     name: nginx
     state: started
     enabled: true
```

Pushing code or files to github repository through Ansible playbook:

```
---
- hosts: localhost
 connection: local
 tasks:
   - name: Ensuring git is installed
     become: yes
     apt:
       name: git
       state: present
```

```
  - name: Clone or update the repository
    git:
      repo: https://github.com/Priyam-Chowdhury/Ansible_push_Calsoft.git
      dest: /home/priyam/Desktop/playbooks/Ansible_push_Calsoft
      clone: yes
      update: yes

  - name: Copy JAR file to the repository directory
    copy:
      src: /home/priyam/Desktop/playbooks/my-app.jar
      dest:
/home/priyam/Desktop/playbooks/Ansible_push_Calsoft/my-app.jar

  - name: Stage changes for commit
    command: chdir=/home/priyam/Desktop/playbooks/Ansible_push_Calsoft
git add my-app.jar

  - name: Commit changes
    command: chdir=/home/priyam/Desktop/playbooks/Ansible_push_Calsoft
git commit -m "Add my-app.jar"

  - name: Push changes to the repository
    command: chdir=/home/priyam/Desktop/playbooks/Ansible_push_Calsoft
git push origin main
```

Ansible register command:

The output of a task can be captured and saved in a variable in Ansible using the register keyword. This variable can then be used in other tasks within the same playbook. This makes it possible to retrieve comprehensive data, including the standard output, error messages, and execution status of the command. Our playbooks can become more dynamic and flexible by saving task outcomes in a variable that allows you to execute additional actions or conditional logic based on the data that was gathered. The registered variable, for instance, can be used to retrieve and show the result of a command or script after it has been performed, as well as to handle any issues that may have occurred.