ANSIBLE

IAC AUTOMATION TOOL

WHAT IS INFRASTRUCTURE AS CODE (IAC)?

- IaC is a practice of managing and provisioning computing infrastructure through machine-readable configuration files.
- It eliminates the need for physical hardware configuration or interactive configuration tools.
- IaC ensures consistency in infrastructure management.
- It enables scalability of infrastructure.
- IaC promotes repeatability, making deployments reliable and predictable.

WHAT IS ANSIBLE?

- Ansible is an open-source automation tool for IT tasks:
  - Configuration management.
  - Application deployment.
  - Infrastructure orchestration.
- It plays a key role in the IaC landscape.
- Features simplicity and agentless architecture.
- Favored by DevOps engineers and IT administrators

Ansible plays a pivotal role in implementing Infrastructure as Code (IaC) by providing a framework to define, manage, and automate infrastructure in a consistent and repeatable manner. Below are the key points that highlight Ansible's strengths:

- Declarative Approach:
  - Ansible allows infrastructure to be defined using a declarative language (YAML), meaning users describe the desired state of the infrastructure, rather than specifying the steps to achieve that state.
- Version Control and Repeatability:
  - Ansible configurations files, can be stored in version control systems like Git, allowing for tracking changes and managing stages effectively.
- Idempotency:
  - Ansible’s idempotency ensures that running the same playbook multiple times will have the same effect, regardless of the current state of the infrastructure.

- Cross-Environment Consistency:
  - Ansible supports the use of playbooks and roles that can be reused across different environments, ensuring infrastructure is provisioned uniformly.
- Scalability and Flexibility:
  - Ansible allows users to define infrastructure and application configurations for various platforms, whether cloud-based (AWS, Azure, GCP), on-premise servers, or hybrid environments.
- Consistency in Infrastructure Management:
  - Ansible makes it possible to apply infrastructure updates and patches consistently across multiple servers, ensuring that configurations are aligned and minimizing the risk of configuration drift.

HOW INSTALL ANSIBLE

Terminal

```
$ sudo apt update
$ sudo apt install ansible
$ ansible --version
```

ANSIBLE IMPORTANT FILES

Inventory File

- The inventory file, also named hosts file, in Ansible is a configuration file that defines the hosts and groups of hosts on which Ansible commands, modules, or playbooks are executed. It provides Ansible with the information about the target systems, such as their IP addresses, hostnames, and any necessary variables

hosts

```
# Group of web servers
[web]
web1.example.com
web2.example.com ansible_host=192.168.1.2 ansible_user=ubuntu
ansible_ssh_private_key_file=/path/to/key
# Group of database servers
[db]
db1.example.com
# Group variables (applied to all hosts in the group)
[web:vars]
ansible_user=root
[db:vars]
ansible_user=admin
```

Example of ansible inventory file

- Here is a list of important attributes (variables) that can be used in an Ansible inventory file to configure hosts:
  - ansible_host
    - Specifies the IP address of the target host.
    - Example: ansible_host=192.168.1.10
  - ansible_port
    - Defines the SSH port for the connection
    - Example: ansible_port=2222
  - ansible_user
    - Specifies the username to use for SSH connections.
    - Example: ansible_user=ubuntu
  - ansible_password
    - Specifies the password to use for SSH connections.
    - Example: ansible_password=yourpassword
  - ansible_ssh_private_key_file
    - Path to the SSH private key file used for authentication.
    - Example: ansible_ssh_private_key_file=/path/to/private_key.pem

Playbook File

- An Ansible playbook is a YAML file that defines a set of tasks to be executed on remote hosts. It provides a structured way to describe the desired state of a system, including configurations, deployments, and orchestration of processes.

Components of a Playbook :

I - Plays

Each play defines:

- hosts: Specifies the target hosts (from the inventory file).
- tasks: A list of actions to be executed.
- name: Optional description of the play.

Playbook

```
- name: Playbook
    hosts: webservers
    become: true
    tasks:
      #Task_code
```

Exemple of Play structure

2 - Tasks

Tasks define the actions performed on the target hosts using Ansible modules. Each task has:

- name: A human-readable name describing the task.
- Module name: The actual action to perform (e.g., apt, copy, file).

Playbook

```
tasks:
  - name: Create a directory
    file:
      path: /opt/mydir
      state: directory
```

Exemple of Task structure

3 - Variables

Variables allow you to parameterize your playbook for reusability. You can define variables:

- Inline in the playbook.
- In separate variable files.

Playbook

```
tasks:
  - name: Install a package
    apt:
      name: "{{ package_name }}"
      state: present
```

Exemple of Variable structure

4 - Roles

Roles allow you to organize playbooks into reusable components, including tasks, variables, files, templates, and handlers.

Playbook

```
- name: Apply webserver configuration
    hosts: webservers
    roles:
        - apache
```

Exemple of Role structure

EXAMPLE : DEPLOYING NGINX APPLICATION

Playbook

Create a file named deploy_nginx.yml:

Playbook

```
- name: Deploy NGINX on webservers
  hosts: webservers
  become: true  # Run tasks with sudo privileges
  tasks:
    - name: Update apt cache
      apt:
        update_cache: yes

    - name: Install NGINX
      apt:
        name: nginx
        state: present
```

Playbook

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

- name: Deploy NGINX configuration
  copy:
    src: files/nginx.conf
    dest: /etc/nginx/nginx.conf
    owner: root
    group: root
    mode: 0644
  notify: Restart NGINX

handlers:
  - name: Restart NGINX
    service:
      name: nginx
      state: restarted
```

- **Tasks:**

  1- Updates the apt cache.

  2- Installs the nginx package.

  3- Ensures the NGINX service is started and enabled.

  4- Copies a custom NGINX configuration file.

- **Handlers:**

  Restart NGINX when the configuration file changes.

Inventory file

Create a file named hosts:

hosts

```
[webservers]
  server1 ansible_host=192.168.1.101
ansible_user=ubuntu
ansible_ssh_private_key_file=~/.ssh/id_rsa
  server2 ansible_host=192.168.1.102
ansible_user=ubuntu
ansible_ssh_private_key_file=~/.ssh/id_rsa
```

Running the Playbook

Terminal

```
$ ansible-playbook -i hosts deploy_nginx.yml
```

After running the playbook NGINX will be installed on all servers in the webservers group, configured to run and start automatically on boot. A custom configuration file will be deployed, and any changes to the configuration will trigger an automatic restart of the NGINX service

THANK YOU

FOR YOUR

ATTENTION