Experiment No : 11

Aim : Automation using Ansible: Spin up a new Linux VM using Ansible playbook

Ansible

Ansible is a software tool that provides simple but powerful automation for cross-platform computer support. It is primarily intended for IT professionals, who use it for application deployment, updates on workstations and servers, cloud provisioning, configuration management, intra-service orchestration, and nearly anything a systems administrator does on a weekly or daily basis. Ansible doesn't depend on agent software and has no additional security infrastructure, so it's easy to deploy.

Because Ansible is all about automation, it requires instructions to accomplish each job. With everything written down in simple script form, it's easy to do version control. The practical result of this is a major contribution to the "infrastructure as code" movement in IT: the idea that the maintenance of server and client infrastructure can and should be treated the same as software development, with repositories of self-documenting, proven, and executable solutions capable of running an organization regardless of staff changes.

While Ansible may be at the forefront of automation, systems administration, and DevOps, it's also useful to everyday users. Ansible allows you to configure not just one computer, but potentially a whole network of computers at once, and using it requires no programming skills. Instructions written for Ansible are human-readable. Whether you're entirely new to computers or an expert, Ansible files are easy to understand.

How Ansible works

In Ansible, there are two categories of computers: the *control node* and *managed nodes*. The control node is a computer that runs Ansible. There must be at least one control node, although a backup control node may also exist. A managed node is any device being managed by the control node.

Ansible works by connecting to nodes (clients, servers, or whatever you're configuring) on a network, and then sending a small program called an Ansible *module* to that node. Ansible executes these modules over SSH and removes them when finished. The only requirement for this interaction is that your Ansible control node has login access to the managed nodes. SSH keys are the most common way to provide access, but other forms of authentication are also supported.

What Ansible does

The term *Ansible modules* sounds complex, but most of the complexity is handled by Ansible and not the user. An Ansible module is written to be a model of the desired state of a system, meaning that each module defines what should be true on any given managed node. For instance, if a systems administrator decides that all workstations in an organization should have LibreOffice version X.Z installed, then it's up to Ansible's packaging module to determine whether each node has LibreOffice X.Z on it. Should Ansible find a managed node with LibreOffice X.Y installed, then it detects the operating system and runs the necessary routine to update it to LibreOffice X.Z. In this way, every workstation in an organization can be updated overnight with the software supported by the IT department.

Ansible Features

1. Configuration Management

Ansible is designed to be very simple, reliable, and consistent for configuration management. If you're already in IT, you can get up and running with it very quickly. Ansible configurations are simple data descriptions of infrastructure and are both readable by humans and parsable by machines. All you need to start managing systems is a password or an SSH (Secure Socket Shell, a network protocol) key. An example of how easy Ansible makes configuration management: If you want to install an updated version of a specific type of software on all the machines in your enterprise, all you have to do is write out all the IP addresses of the nodes (also called remote hosts) and write an Ansible playbook to install it on all the nodes, then run the playbook from your control machine.

2. Application Deployment

Ansible lets you quickly and easily deploy multitier apps. You won't need to write custom code to automate your systems; you list the tasks required to be done by writing a playbook, and Ansible will figure out how to get your systems to the state you want them to be in. In other words, you won't have to configure the applications on every machine manually. When you run a playbook from your control machine, Ansible uses SSH to communicate with the remote hosts and run all the commands (tasks).

3. Orchestration

As the name suggests, orchestration involves bringing different elements into a beautifully run whole operation—similar to the way a musical conductor brings the notes produced by all the different instruments into a cohesive artistic work. For example, with application deployment, you need to manage not just the front-end and backend services but the databases, networks, storage, and so on. You also need to make sure that all the tasks are handled in the proper order. Ansible uses automated workflows, provisioning, and more to make orchestrating tasks easy. And once you've defined your infrastructure using the Ansible playbooks, you can use that same orchestration wherever you need to, thanks to the portability of Ansible playbooks.

4. Security and Compliance

As with application deployment, sitewide security policies (such as firewall rules or locking down users) can be implemented along with other automated processes. If you configure the security details on the control machine and run the associated playbook, all the remote hosts will automatically be updated with those details. That means you won't need to monitor each machine for security compliance continually manually. And for extra security, an admin's user ID and password aren't retrievable in plain text on Ansible.

5. Cloud Provisioning

The first step in automating your applications' life cycle is automating the provisioning of your infrastructure. With Ansible, you can provision cloud platforms, virtualized hosts, network devices, and bare-metal servers.

Ansible Terms:

- •**Controller Machine**: The machine where Ansible is installed, responsible for running the provisioning on the servers you are managing.
- •**Inventory**: An initialization file that contains information about the servers you are managing.
- •**Playbook**: The entry point for Ansible provisioning, where the automation is defined through tasks using YAML format.
- •**Task**: A block that defines a single procedure to be executed, e.g. Install a package.
- •**Module**: A module typically abstracts a system task, like dealing with packages or creating and changing files. Ansible has a multitude of built-in modules, but you can also create custom ones.
- •**Role**: A pre-defined way for organizing playbooks and other files in order to facilitate sharing and reusing portions of a provisioning.
- •**Play**: A provisioning executed from start to finish is called a play. In simple words, execution of a playbook is called a play.
- •**Facts**: Global variables containing information about the system, like network interfaces or operating system.
- •**Handlers**: Used to trigger service status changes, like restarting or stopping a service.

Benefits of Ansible

- ●**Free**: Ansible is an open-source tool.
- ●**Very simple to set up and use**: No special coding skills are necessary to use Ansible's playbooks (more on playbooks later).
- ●**Powerful**: Ansible lets you model even highly complex IT workflows.
- ●**Flexible**: You can orchestrate the entire application environment no matter where it's deployed. You can also customize it based on your needs.
- ●**Agentless**: You don't need to install any other software or firewall ports on the client systems you want to automate. You also don't have to set up a separate management structure.

- **Efficient**: Because you don’t need to install any extra software, there’s more room for application resources on your server.

Ansible playbooks

Playbooks are the files where Ansible code is written. Playbooks are written in YAML format. YAML stands for Yet Another Markup Language. **Playbooks** are one of the core features of Ansible and tell Ansible what to execute. They are like a to-do list for Ansible that contains a list of tasks.

Playbooks contain the steps which the user wants to execute on a particular machine. Playbooks are run sequentially. Playbooks are the building blocks for all the use cases of Ansible.

Playbook Structure

Each playbook is an aggregation of one or more plays in it. Playbooks are structured using Plays. There can be more than one play inside a playbook.

The function of a play is to map a set of instructions defined against a particular host.

YAML is a strict typed language; so, extra care needs to be taken while writing the YAML files. There are different YAML editors but we will prefer to use a simple editor like notepad++. Just open notepad++ and copy and paste the below yaml and change the language to YAML (Language → YAML).

While modules provide the means of accomplishing a task, the way you use them is through an Ansible *playbook*. A playbook is a configuration file written in YAML that provides instructions for what needs to be done in order to bring a managed node into the desired state. Playbooks are meant to be simple, human-readable, and self-documenting. They are also *idempotent*, meaning that a playbook can be run on a system at any time without having a negative effect upon it. If a playbook is run on a system that's already properly configured and in its desired state, then that system should still be properly configured after a playbook runs.A playbook can be very simple, such as this one that installs, as a privileged user, the Apache HTTP server on any node in an IT department's *webservers* group:

```
---
- name: Apache server installed
  hosts: webservers
  become: yes
```

Playbooks can also be very complex, with conditionals and variables. However, because most of the real work is done by Ansible modules, playbooks remain brief, readable, and clear even though they can orchestrate entire networks of managed nodes.

The Different YAML Tags

Let us now go through the different YAML tags. The different tags are described below −

name

This tag specifies the name of the Ansible playbook. As in what this playbook will be doing. Any logical name can be given to the playbook.

hosts

This tag specifies the lists of hosts or host group against which we want to run the task. The hosts field/tag is mandatory. It tells Ansible on which hosts to run the listed tasks. The tasks can be run on the same machine or on a remote machine. One can run the tasks on multiple machines and hence hosts tag can have a group of hosts' entry as well.

vars

Vars tag lets you define the variables which you can use in your playbook. Usage is similar to variables in any programming language.

tasks

All playbooks should contain tasks or a list of tasks to be executed. Tasks are a list of actions one needs to perform. A tasks field contains the name of the task. This works as the help text for the user. It is not mandatory but proves useful in debugging the playbook. Each task internally links to a piece of code called a module. A module that should be executed, and arguments that are required for the module you want to execute.