LTI

Celebrating 20 Years

Ansible Documentation

Mukul Katiyar
PS No. 10671391

A Larsen & Toubro Group Company

CONTEXT:

Ansible is an open-source IT engine which automates the IT tools such as intra service orchestration, application deployment, cloud provisioning, etc.

Ansible is easy to deploy because it does not use any **agents** or **custom security** infrastructure on the client-side, and by pushing modules to the clients. These modules are executed locally on the client-side, and the output is pushed back to the Ansible server. It can easily connect to clients using **SSH-Keys**, simplifying though the whole process. Client details, such as **hostnames** or **IP addresses** and **SSH ports**, are stored in the files, which are called inventory files. If you created an inventory file and populated it, then Ansible can use it.

Ansible uses the playbook to describe automation jobs, and playbook, which uses simple language, i.e., **YAML**. YAML is a human-readable data serialization language & commonly used for configuration files, but it can be used in many applications where data is being stored.

Ansible is designed for multi-tier deployment. Ansible does not manage one system at a time, and it models IT infrastructure by describing all of your systems are interrelated. Ansible is entirely agentless, which means Ansible works by connecting your nodes through **SSH** (by default). Ansible gives the option to you if you want another method for the connection like **Kerberos**.

Ansible pushes small programs after connecting to your nodes which are known as "**Ansible Modules**". Ansible runs that module on your nodes and removes them when finished. Ansible manages the inventory in simple text files (These are the host's files). Ansible uses the host file where one can group the hosts and can control the actions on a specific group in the playbooks.

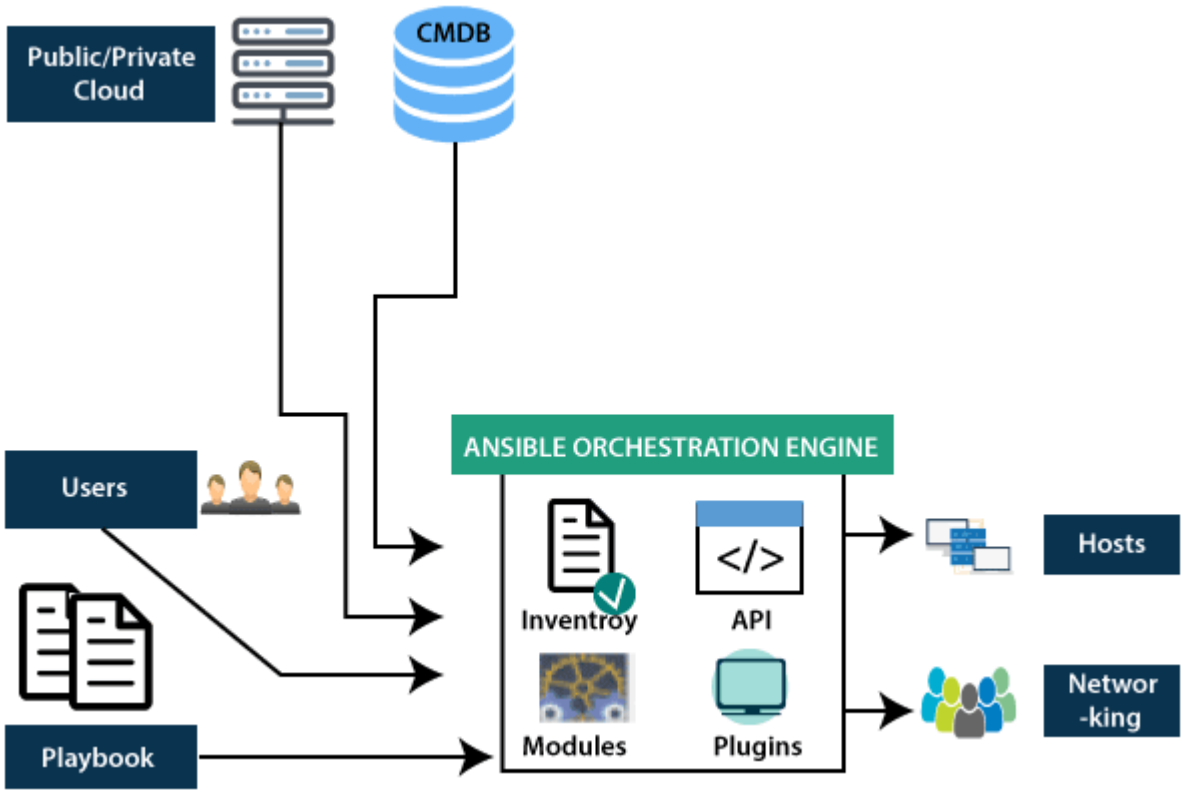
Features :

- In this, It uses no extra functionality and cost like no agents and no extra custom security infrastructure, hence it is easy to deploy.

- It uses a very simple language called YAML (Yet Another Markup Language) in the form of Ansible Playbooks and you can configure it as per your requirement, and it helps describe the automation jobs in a way that looks like basic English.
- The Ansible Automation Engine has a direct interaction with the users who write playbooks and also interacts with cloud services and the Configuration Management Database (CMDB).

Architecture Components:

The Ansible orchestration engine interacts with a user who is writing the Ansible playbook to execute the Ansible orchestration and interact along with the services of private or public cloud and configuration management database. You can show in the below diagram, such as:



Inventory

Inventory is lists of nodes or hosts having their IP addresses, databases, servers, etc. which are need to be managed.

API's

The Ansible API's works as the transport for the public or private cloud services.

Modules

Ansible connected the nodes and spread out the Ansible modules programs. Ansible executes the modules and removed after finished. These modules can reside on any machine; no database or servers are required here. You can work with the chose text editor or a terminal or version control system to keep track of the changes in the content.

Plugins

Plugins is a piece of code that expends the core functionality of Ansible. There are many useful plugins, and you also can write your own.

Playbooks

Playbooks consist of your written code, and they are written in YAML format, which describes the tasks and executes through the Ansible. Also, you can launch the tasks synchronously and asynchronously with playbooks.

Hosts

In the Ansible architecture, hosts are the node systems, which are automated by Ansible, and any machine such as RedHat, Linux, Windows, etc.

Networking

Ansible is used to automate different networks, and it uses the simple, secure, and powerful agentless automation framework for IT operations and development. It uses a type of data model which separated from the Ansible automation engine that spans the different hardware quite easily.

Cloud

A cloud is a network of remote servers on which you can store, manage, and process the data. These servers are hosted on the internet and storing the data remotely rather than the local server. It just launches the resources and instances on the cloud, connect them to the servers, and you have good knowledge of operating your tasks remotely.

CMDB

CMDB is a type of repository which acts as a data warehouse for the IT installations.

SOLUTION:

To run Ansible we first need to install Ansible on our WorkstationVM. Ansible on this VM establishes connection with other hosts via SSH. We fists configure SSH connection between the workstation and the host VMs. After SSH is configured we install Ansible and we are good to go to perform automation and maintainence tasks on our host VMs. The following steps highlight the process.

Step1- Make sure OpenSSH server is installed on workstation and host VMs. Use following commands to install OpenSSH.

Type `sudo apt-get install openssh-server`

Enable the ssh service by typing `sudo systemctl enable ssh`

Start the ssh service by typing `sudo systemctl start ssh`

Test it by login into the system using `ssh user@server-name`

If the VM setup in the documentation has been followed properly we will be able to connect to the host machine using SSH. We can replace user@server-name in the login command with the IP address of the host we wish to connect to. We can check the IP configuration of each host by typing command ip a in the terminal. Ensure that we are able to connect to each host machine using SSH. Once this is done we will set SSH keys for Ansible to make our connection more secure. This step is not compulsory but a good practice.

Step 2- SSH keys are used as login credentials, often in place of simple clear text passwords. They work in pairs: we always have a **public** and a **private** key. The private key must remain on the local computer which acts as the client: it is used to **decrypt** information and it must never be shared. The public key, on the other hand, is used to encrypt data and must be copied on the remote server (its content is copied in the `~/.ssh/authorized_keys` file in the $HOME directory of the user we login as on the server - we will see how to perform such operation in the course of this tutorial).

The ability to use ssh-keys as login credentials must be allowed server-side by the system administrator, by setting the `PubkeyAuthentication` option to `yes` in the `/etc/ssh/sshd.config` file. Both clear text passwords and public keys can be allowed as authentication methods at the same time, or, for example, one could decide to allow access only via public keys.

To generate a SSH key type `ssh keygen -t ed25519-f ~/.ssh/Ansible`

Copy the SSH key to each of the hosts using `ssh-copy-id -I ~/.ssh/Ansible.pub 192.168.0.9`

After running the above command we will get a prompt that the key was added to the server. We may need to enter the sudo password for the host to add the key.

Step 3- We install Ansible in this step. A great manual to install Ansible can be found in the following link.

Install Ansible in Ubuntu

Step 4- The various tasks with Ansible have been completed with explanation of the codes in the comments in the files. The link of the Github Repository for the same is Ansible Github.