what is ansible ?

Ansible is an open source automation platform. Ansible can help you with configuration management, application deployment, task automation and also IT orchestration. It is very simple to setup, efficient and more powerful.

Ansible is available for free and run on linux, Mac. Aside from the free offering, Ansible also has an enterprise product called Ansible Tower.

Ansible is developed in Python language.
It is a software tool. It is useful while deploying any application using ssh without any downtime. Using this tool one can manage and configure software applications very easily.
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.

A significant advantage is that even the IT infrastructure support guys can read and understand the playbook and debug if needed.

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.

Ansible History

Here are some essential points from the history of Ansible, such as:

- **Michael DeHaan** developed Ansible, and the Ansible project began in **February 2012**.
- The creator of **Cobbler** and **Func** is also the controller of the **Fedora Unified** network.
- **RedHat** acquired the Ansible tool in 2015.
- Ansible is included as part of the **Fedora distribution** of the Linux.
- Ansible is also available for **RedHat Enterprise Linux, Debian, CentOS, Oracle Linux,** and **Scientific Linux** via Extra Packages for Enterprise Linux **(EPEL)** and **Ubuntu** as well as for other operating systems.

why do we use ansible ?

Here are some important reasons for using Ansible, such as:

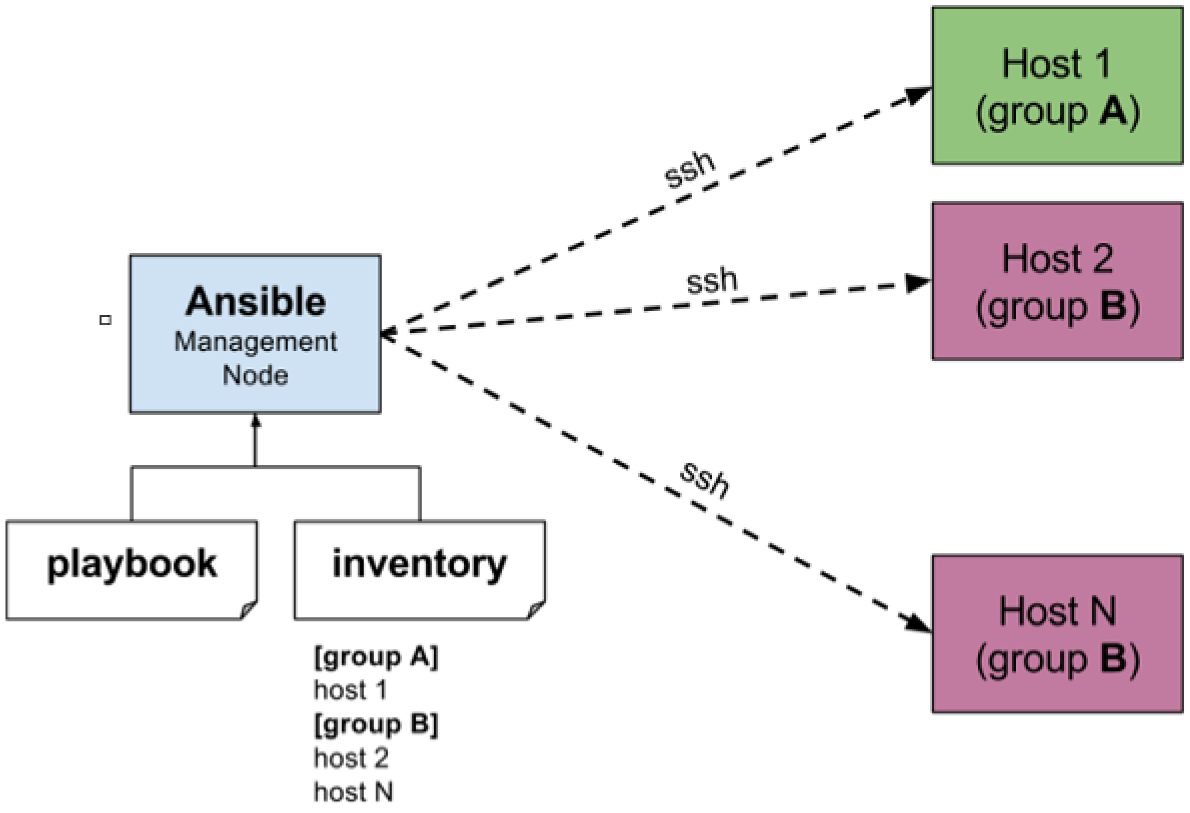
- We have lot of automations tools available in the market namely puppet, chef and saltstack. Obviously, each tools has its own features and functionality.
  Developed in Python. Python is that it is inbuilt into most Unix and Linux deployments nowadays, so getting up and running is quicker.
- Ansible is free to use by everyone.
- Ansible is very consistent and lightweight, and no constraints regarding the operating system or underlying hardware are present. or lightweight and quick deployment. Ofcourse its lightweight and its very fast to deployment, because it uses the SSH, doesn't have master client communication compared to other tools.
- It is very secure due to its agentless capabilities and open **SSH** security features.

- Ansible does not need any special system administrator skills to install and use it.
- Ansible uses YAML Syntax in configuration files.
- Larger community. Ansible has a large and engaged community of users who can help answer your questions.
- Ansible is a helpful tool that allows you to create groups of machines, describe how these machines should be configured or what actions should be taken on them. Ansible issues all commands from a central location to perform these tasks.
- No other client software is installed on the node machines. It uses SSH to connect to the nodes.
- Ansible only needs to be installed on the control machine (the machine from which you will be running commands) which can even be your laptop. It is a simple solution to a complicated problem.

How Ansible Works?

The picture given below shows the working of Ansible.

Ansible works by connecting to your nodes and pushing out small programs, called "**Ansible** modules" to them. **Ansible** then executes these modules (over SSH by default), and removes them when finished. Your library of modules can reside on any machine, and there are no servers, daemons, or databases required.



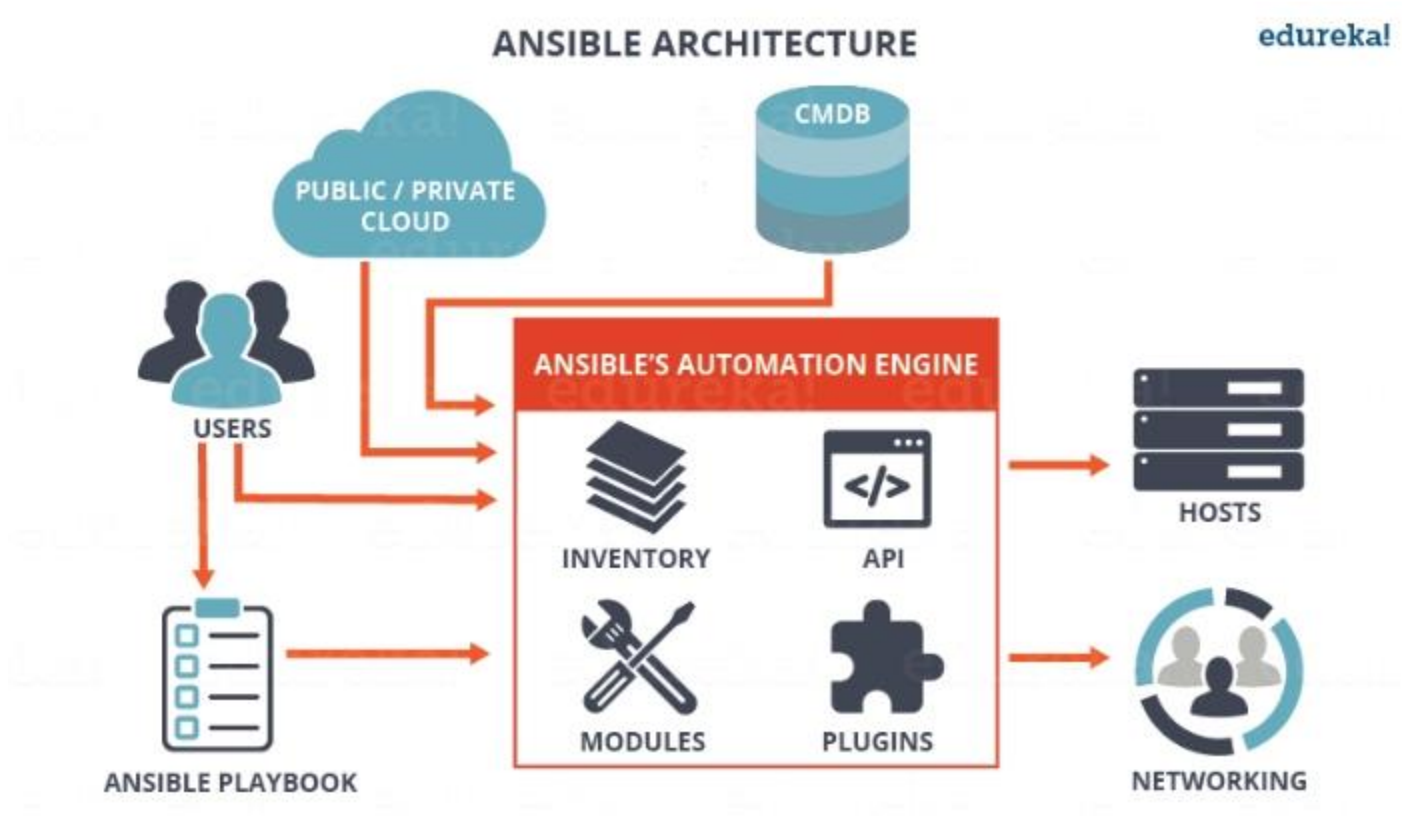
In the above image, the Management Node is the controlling node that controls the entire execution of the playbook. The inventory file provides the list of hosts where the Ansible modules need to be run. The Management Node makes an SSH connection and executes the small modules on the host's machine and install the software.

Ansible removes the modules once those are installed so expertly. It connects to the host machine executes the instructions, and if it is successfully installed, then remove that code in which one was copied on the host machine.

The management node in the above picture is the controlling node (managing node) which controls the entire execution of the playbook. It's the node from which you are running the installation. The inventory file provides the list of hosts where the Ansible modules needs to be run and the management node does a SSH connection and executes the small modules on the hosts machine and installs the product/software.**Beauty** of Ansible is that it removes the modules once those are installed so effectively it connects to host machine , executes the instructions and if it’s successfully installed removes the code which was copied on the host machine which was executed.

What is Ansible & its Architecture?

Ansible architecture is fairly straightforward. Refer to the diagram below to understand the Ansible architecture:



As you can see, in the diagram above, the Ansible automation engine has a direct interaction with the users who write playbooks to execute the Ansible Automation engine. It also interacts with cloud services and Configuration Management Database (CMDB).

The Ansible Automation engine consists of:

- Inventories: Ansible inventories are lists of hosts (nodes) along with their IP addresses, servers, databases etc. which needs to be managed. Ansible then takes action via a transport – SSH for UNIX, Linux or Networking devices and WinRM for Windows system.

- **APIs:** APIs in Ansible are used as transport for Cloud services, public or private.
- **Modules:** Modules are executed directly on remote hosts through playbooks. The modules can control system resources, like services, packages, or files (anything really), or execute system commands. Modules do it by acting on system files, installing packages or making API calls to the service network. There are over 450 Ansible-provided modules that automate nearly every part of your environment. For e.g.
  - Cloud Modules like *cloudformation* which creates or deletes an AWS cloud formation stack;
  - Database modules like *mssql_db* which removes MYSQL databases from remote hosts.
- **Plugins**: Plugins allows to execute Ansible tasks as a job build step. Plugins are pieces of code that augment Ansible's core functionality. Ansible ships with a number of handy plugins, and you can easily write your own. For example,
  - *Action* plugins are front ends to modules and can execute tasks on the controller before calling the modules themselves.
  - *Cache* plugins are used to keep a cache of 'facts' to avoid costly fact-gathering operations.
  - *Callback* plugins enable you to hook into Ansible events for display or logging purposes.

There are a few more components in Ansible Architecture which are explained below:

Networking: Ansible can also be used to automate different networks. Ansible uses the same simple, powerful, and the agentless automation framework IT operations and development are already using. It uses a data model (a playbook or role) that is separate from the Ansible automation engine that easily spans different network hardware.

Hosts: The hosts in the Ansible architecture are just node systems which are getting automated by Ansible. It can be any kind of machine – Windows, Linux, RedHat etc.

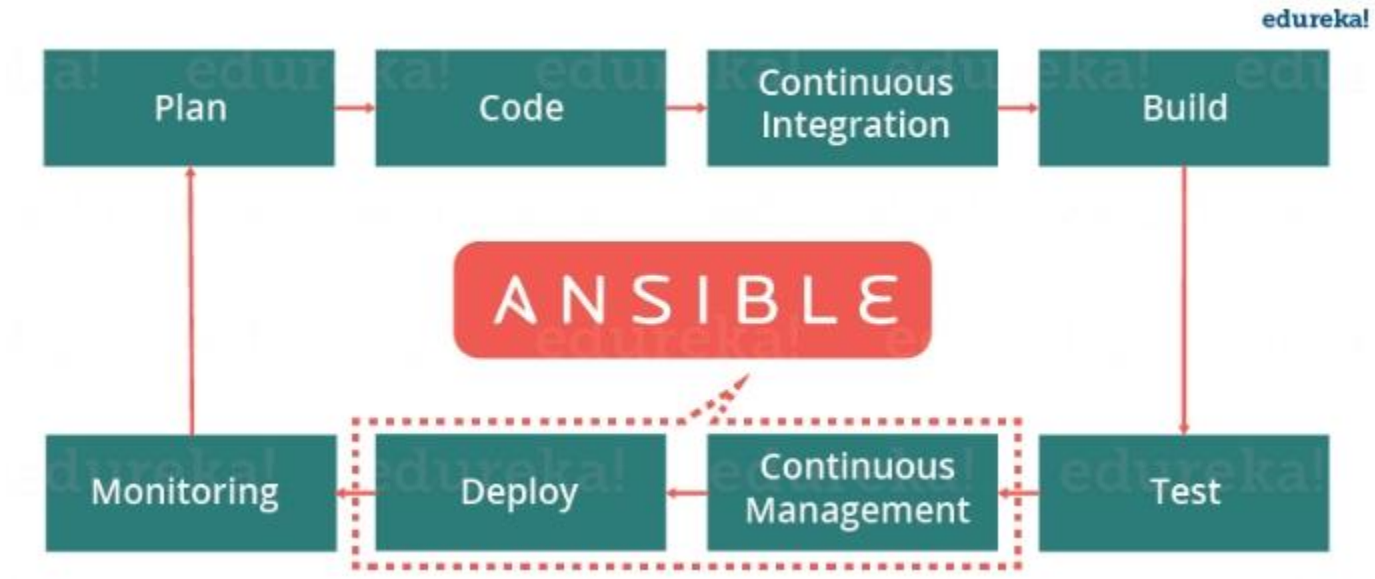
Playbooks: Playbooks are simple files written in YAML format which describes the tasks to be executed by Ansible. Playbooks can declare configurations, but they can also orchestrate the steps of any manual ordered process, even if it contains jump statements. They can launch tasks synchronously or asynchronously.

CMDB : It is a repository that acts as a data warehouse for IT installations. It holds data relating to a collection of IT assets (commonly referred to as configuration items (CI)), as well as to describe relationships between such assets.

Cloud: It is a network of remote servers hosted on the Internet to store, manage, and process data, rather than a local server. You can launch your resources and instances on cloud and connect to your servers.

What is Ansible in DevOps?

In DevOps, as we know development and operations work is integrated. This integration is very important for modern test-driven application design. Hence, Ansible integrates this by providing a stable environment to both development and operations resulting in smooth orchestration. Refer to the image below to see how Ansible fits into DevOps:



Let us discuss now how Ansible manages the entire DevOps infrastructure. When developers begin to think of infrastructure as part of their application i.e as Infrastructure as code (**IaC**), stability and performance become normative. Infrastructure as Code is the process of managing and provisioning computing infrastructure (processes, bare-metal servers, virtual servers, etc.) and their configuration through machine-processable definition files, rather than physical hardware configuration or the use of interactive configuration tools. This is where Ansible automation plays a major role and stands out among its peers.

In DevOps, Sysadmins work tightly with developers, development velocity is improved, and more time is spent doing activities like performance tuning, experimenting, and getting things done, and less time is spent fixing problems. Refer to the diagram below to understand how the tasks of sysadmins and other users are simplified by Ansible.

writing playbooks in ansible :

Playbooks in Ansible are written in YAML format. It is a human-readable data serialization language. It is commonly used for configuration files. It can also be used in many applications where data is being stored.

For Ansible, nearly every YAML file starts with a list. Each item in the list is a list of key/value pairs, commonly called a “hash” or a “dictionary”. So, we need to know how to write lists and dictionaries in YAML.

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

Ansible playbooks tend to be more configuration language than a programming language.

All members of a list are lines beginning at the same indentation level starting with a “- ” (dash and space). More complicated data structures are possible, such as lists of dictionaries or mixed dictionaries whose values are lists or a mix of both.