# Ansible - Introduction

**Ansible** is simple open source IT engine which automates application deployment, intra service orchestration, cloud provisioning and many other IT tools.

Ansible is easy to deploy because it does not use any agents or custom security infrastructure.

Ansible uses playbook to describe automation jobs, and playbook uses very simple language i.e. **YAML** (It's a human-readable data serialization language & is commonly used for configuration files, but could be used in many applications where data is being stored)which is very easy for humans to understand, read and write. Hence the advantage is that even the IT infrastructure support guys can read and understand the playbook and debug if needed (YAML – It is in human readable form).

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

After connecting to your nodes, Ansible pushes small programs called as "Ansible Modules". Ansible runs that modules on your nodes and removes them when finished. Ansible manages your inventory in simple text files (These are the hosts file). Ansible uses the hosts file where one can group the hosts and can control the actions on a specific group in the playbooks.

## Sample Hosts File

This is the content of hosts file −

```
#File name: hosts
#Description: Inventory file for your application. Defines machine
type abc
node to deploy specific artifacts
# Defines machine type def node to upload
metadata.

[abc-node]
#server1 ansible_host = <target machine for DU deployment>
ansible_user = <Ansible
user> ansible_connection = ssh
server1 ansible_host = <your host name> ansible_user = <your unix
user>
ansible_connection = ssh

[def-node]
```

```
#server2 ansible_host = <target machine for artifact upload>
ansible_user = <Ansible user> ansible_connection = ssh
server2 ansible_host = <host> ansible_user = <user>
ansible_connection = ssh
```

## What is Configuration Management

Configuration management in terms of Ansible means that it maintains configuration of the product performance by keeping a record and updating detailed information which describes an enterprise's hardware and software.

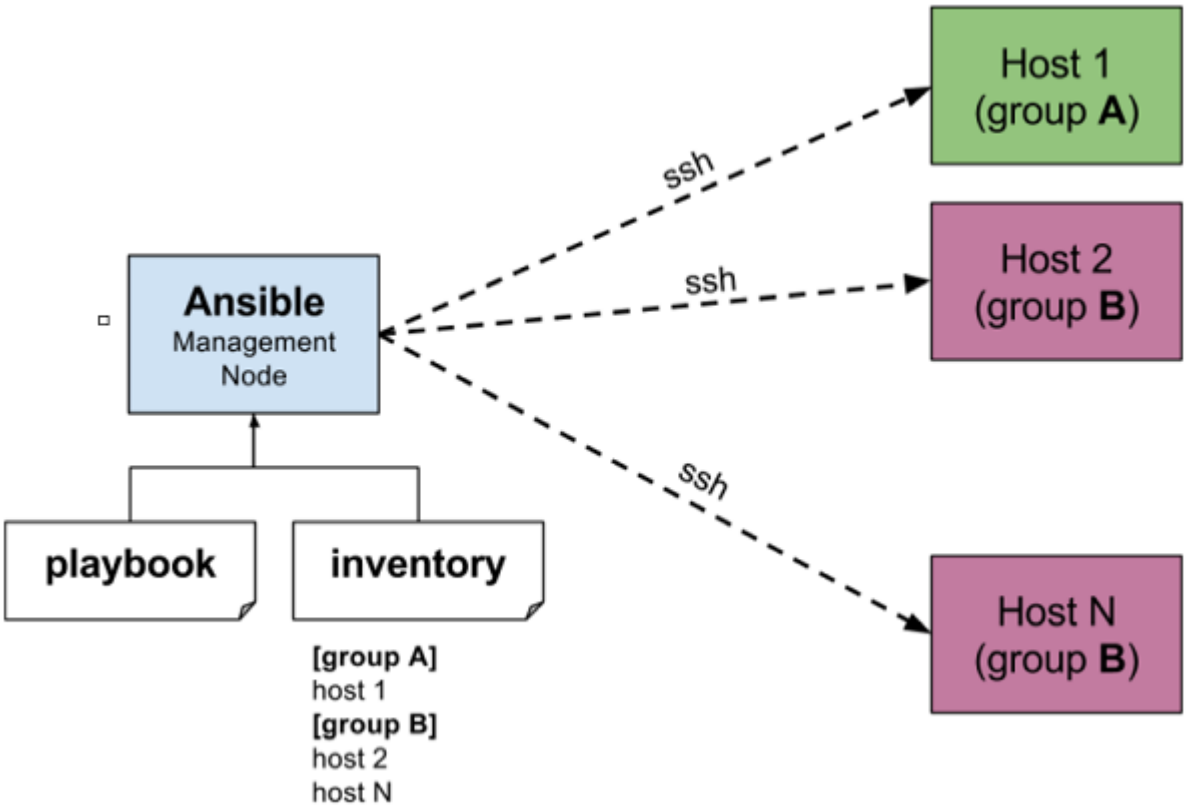
Such information typically includes the exact versions and updates that have been applied to installed software packages and the locations and network addresses of hardware devices. For e.g. If you want to install the new version of **WebLogic/WebSphere** server on all of the machines present in your enterprise, it is not feasible for you to manually go and update each and every machine.

You can install WebLogic/WebSphere in one go on all of your machines with Ansible playbooks and inventory written in the most simple way. All you have to do is list out the IP addresses of your nodes in the inventory and write a playbook to install WebLogic/WebSphere. Run the playbook from your control machine & it will be installed on all your nodes.

## 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.




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.

## Installation Process

Mainly, there are two types of machines when we talk about deployment −

- **Control machine** − Machine from where we can manage other machines.
- **Remote machine** − Machines which are handled/controlled by control machine.

There can be multiple remote machines which are handled by one control machine. So, for managing remote machines we have to install Ansible on control machine.

### Control Machine Requirements

Ansible can be run from any machine with Python 2 (versions 2.6 or 2.7) or Python 3 (versions 3.5 and higher) installed.

**Note** − Windows does not support control machine.

By default, Ansible uses **ssh** to manage remote machine.

Ansible does not add any database. It does not require any daemons to start or keep it running. While managing remote machines, Ansible **does not** leave any software installed or running on them. Hence, there is no question of how to upgrade it when moving to a new version.

Ansible can be installed on control machine which have above mentioned requirements in different ways. You can install the latest release through Apt, yum, pkg, pip, OpenCSW, pacman, etc.

Ansible uses YAML syntax for expressing Ansible playbooks. This chapter provides an overview of YAML. Ansible uses YAML because it is very easy for humans to understand, read and write when compared to other data formats like XML and JSON.

Every **YAML** file optionally starts with "---" and ends with "...".

# Understanding YAML

In this section, we will learn the different ways in which the YAML data is represented.

## key-value pair

YAML uses simple key-value pair to represent the data. The dictionary is represented in key: value pair.

**Note** − There should be space between : and value.

## Example: A student record

```
--- #Optional YAML start syntax
james:
   name: james john
   rollNo: 34
   div: B
   sex: male
… #Optional YAML end syntax
```

## Abbreviation

You can also use abbreviation to represent dictionaries.

## Example

```
James: {name: james john, rollNo: 34, div: B, sex: male}
```

## Representing List

We can also represent List in YAML. Every element(member) of list should be written in a new line with same indentation starting with "- " (- and space).

### Example

```
---
countries:
   - America
   - China
   - Canada
   - Iceland
…
```

### Abbreviation

You can also use abbreviation to represent lists.

### Example

```
Countries: ['America', 'China', 'Canada', 'Iceland']
```

### List inside Dictionaries

We can use list inside dictionaries, i.e., value of key is list.

### Example

```
---
james:
   name: james john
   rollNo: 34
   div: B
   sex: male
   likes:
      - maths
      - physics
      - english
…
```

### List of Dictionaries

We can also make list of dictionaries.

### Example

```
---
- james:
   name: james john
   rollNo: 34
      div: B
   sex: male
   likes:
      - maths
```

```
      - physics
      - english

- robert:
      name: robert richardson
      rollNo: 53
      div: B
      sex: male
   likes:
      - biology
      - chemistry
…
```

YAML uses “|” to include newlines while showing multiple lines and “>” to suppress newlines while showing multiple lines. Due to this we can read and edit large lines. In both the cases intendentation will be ignored.

We can also represent **Boolean** (True/false) values in YAML. where **boolean** values can be case insensitive.

## Example

```
---
- james:
   name: james john
   rollNo: 34
   div: B
   sex: male
   likes:
      - maths
      - physics
      - english

   result:
      maths: 87
      chemistry: 45
      biology: 56
      physics: 70
      english: 80

   passed: TRUE

   messageIncludeNewLines: |
      Congratulation!!
      You passed with 79%

   messageExcludeNewLines: >
      Congratulation!!
      You passed with 79%
```

## Some common words related to Ansible.

**Service/Server** − A process on the machine that provides the service.

**Machine** − A physical server, vm(virtual machine) or a container.

**Target machine** − A machine we are about to configure with Ansible.

**Task** − An action(run this, delete that) etc managed by Ansible.

**Playbook** − The yml file where Ansible commands are written and yml is executed on a machine.