# Introduction

## What is Automation?

- *Automation* can help avoid the problems caused by manual system administration and Infrastructure management.
- As a system administrator, you can use it to ensure that all your systems are quickly and correctly deployed and configured.
- This allows you to automate the repetitive tasks in your daily schedule, freeing up your time and allowing you to focus on more critical things.
- For your organization, this means you can more quickly roll out the next version of an application or updates to a service.

## ANSIBLE

- The Ansible tool was developed by **Michael DeHaan**, the author of the provisioning server application Cobbler and co-author of the Fedora Unified Network Controller (Func) framework for remote administration.
- In 2018 Ansible 2.4 version was launched and latest version in 2.9
- Ansible is an open source automation platform.
- It is a simple automation language that can perfectly describe an IT application infrastructure in Ansible Playbooks.
- It is also an automation engine that runs Ansible Playbooks.
- Ansible can manage powerful automation tasks and can adapt too many different workflows and environments.
- At the same time, new users of Ansible can very quickly use it to become productive.

## Why Ansible

- **Ansible Is Simple**
  - Ansible Playbooks provide human-readable automation.
  - No special coding skills are required to write them.
  - Playbooks execute tasks in order.

- **Ansible Is Powerful**
  - You can use Ansible to deploy applications, for configuration management, for workflow automation, and for network automation.
  - Any modules that are pushed are removed when Ansible is finished with its tasks.

- **Ansible Is Agentless**
  - Ansible connects to the hosts it manages using OpenSSH or WinRM and runs tasks, by pushing out small programs called *Ansible modules* to those hosts.

        - Because there are no agents and no additional custom security infrastructure, Ansible is more efficient and more secure than other alternatives.

    - **Cross platform support**
        - Ansible provides agentless support for Linux, Windows, UNIX, and network devices, in physical, virtual, cloud, and container environments.

    - **Perfect description of applications:**
        - Every change can be made by Ansible Playbooks, and every aspect of your application environment can be described and documented.

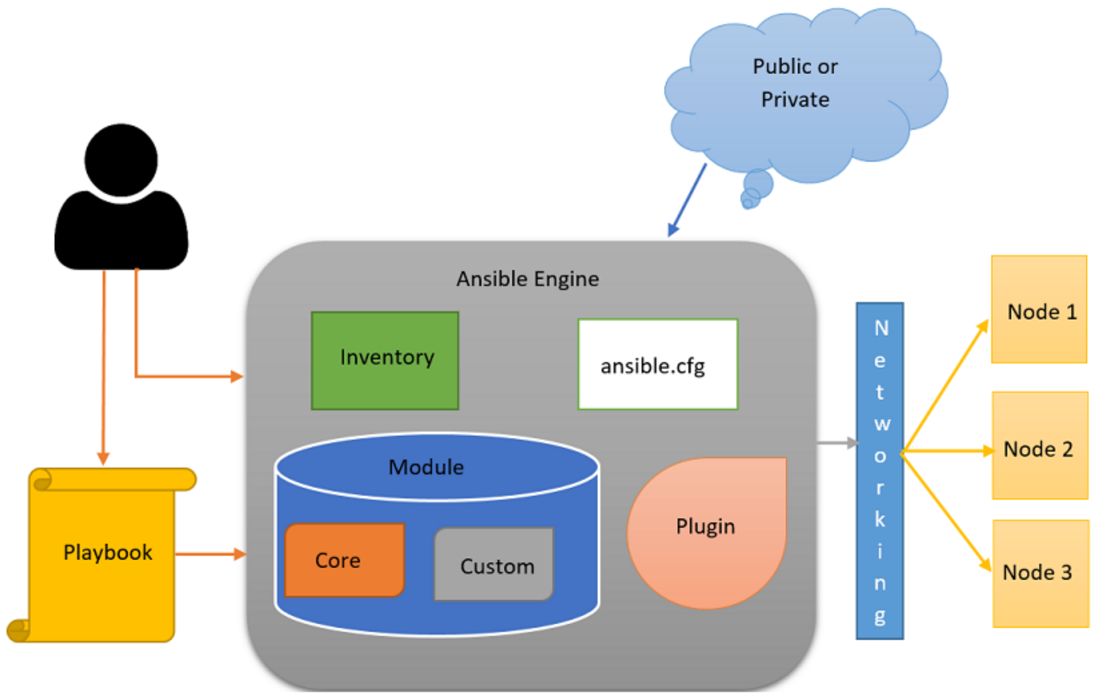
    - **Easy to manage in version control:**
        - Ansible Playbooks and projects are plain text. They can be treated like source code and placed in your existing version control system.

    - **Support for dynamic inventories:**
        - The list of machines that Ansible manages can be dynamically updated.

## ANSIBLE CONCEPTS AND ARCHITECTURE

- There are two types of machines in the Ansible architecture: control nodes and managed hosts.
- Ansible is installed and run from a control node, and this machine also has copies of your Ansible project files.
- A control node could be an administrator's laptop, or from where the Administrator will work.
- Managed hosts are listed in an inventory, which also organizes those systems into groups for easier collective management. Which can be static or dynamic (script/python/external source).
- Ansible users create high-level plays to ensure a host or group of hosts are in a particular state.
- These plays are expressed in YAML format in a text file.
- A file that contains one or more plays is called a playbook.
- Each task runs a module, a small piece of code, with specific arguments.
- Tasks, plays, and playbooks are designed to be idempotent. This means that you can safely run a playbook on the same hosts multiple times.
- Tasks, plays, and playbooks are designed to be *idempotent*.

- Ansible also uses *plug-ins*. Plug-ins are code that you can add to Ansible to extend it and adapt it to new uses and platforms.



**Install Ansible**

**Ansible Control Nodes**

- Ansible is simple to install. The Ansible software only needs to be installed on the control node (or nodes) from which Ansible will be run.
- The control node should be a Linux or UNIX system.
- Microsoft Windows is not supported as a control node, although Windows systems can be managed hosts.
- Python 3 (version 3.5 or later) or Python 2 (version 2.7 or later) needs to be installed on the control node.

**Install Ansible using yum in (Register machine):**

```
# yum search ansible
# subscription-manager repos –list | grep ansible
# subscription-manager repos - -enable ansible-2.8-for-rhel-8-x86_64-rpms
# yum search ansible
# yum install ansible –y
```

**Install Ansible using yum in (non-Register Machine):**

```
# yum install –y https://dl.fedoraproject.org/pub/epel/epel-release-latest-8.noarch.rpm
# yum inatall ansible –y
# ansible --version
```

**Install Ansible from source**

```
# yum install git -y
# mkdir ansible        # in user home location
# mkdir git
# cd git
# git clone - -single-branch - -branch=stable-2.8 https://github.com/ansible/ansible.git
# cd ansible
# source ./hacking/env-setup
# echo $PATH
# vi ~/.bash_profile
        #End of the line
        source ~/git/ansible/hacking/env-setup
        wq:

# cat requirements.txt
# pip2.7 install - -user -r ./requirements.txt
# alternatives - -set python /usr/bin/python3
# ansible {tab tab}
# ansible 127.0.0.1 -m ping
```

--user = the flag is used to tells **Pip (python Package manager)** to install packages in some specific directories within your home directory.
-r = --requirement **Install** from the given requirements file

## Configuration File:

### The Ansible Configuration files

- Possible location of Ansible configuration files (in order processed)
  - ANSIBLE_CONFIG (environment variable)
  - ansible.cgf (in the home directory)
  - ~ / .ansible .cfg (in the home directory)
  - /etc /ansible /ansible.cfg (Default location)
- A configuration file will not automatically load if it is in a world -writable directory.
- Configuration can be set in environment variable.

### Common Ansible Configuration:

- The ansible -config command can be used to view configuration:
- list -Prints all configuration option
- dump -Dumps configuration
- view- View the configuration file

**Commonly used setting**

- inventory- specifies the default inventory file
- roles_path  - set paths to search in  for roles
- forks -specifies  the amount of hosts  configured  by ansible  at the  same time (parallelism):
- ansible _managed- text inserted  into  templates which indicate that file is managed by Ansible and change will be overwritten.

**Lab-2 (Create. Default ansible.cfg)**

```
etc             # mkdir ansible
                # mkdir ansible/roles
                # cd ansible
ansible         # ls /home/ansadmin/git/ansible/examples
                #cp /home/ansadmin/git/ansible/examples/ansible.cfg .
                #cp /home/ansadmin/git/ansible/examples/hosts .
                # ls -l
                # vi ansible.cfg
```

**Lab-2 (Create custom ansible.cfg in user home location)**

```
$pwd
$ vi ansible.cfg
[defaults]

interpreter_python = auto
inventory = /home/ansadmin/ansible/inventory/inv.ini
roles = /etc/ansible/roles
```

Note: if you are not setting all parameter, remaining will take from default location
Also interpreter in default location- /etc/ansible/ansible.cfg