 **RED HAT** **ANSIBLE**

**Environment**

An environment means the underlying infrastructure, the configurations and various dependencies. The underlying infra defines in which place the application would run, the configurations needed by the application to run and how other external dependencies will associate with the application. The configuration will decide how the application will fit and perform in the infrastructure that it is deployed in. The dependencies would mean the various libraries and services from other modules or systems which are used by the application.

**Server Configuration Management (SCM)**

When there are more than one systems involved in developing a software, maintaining the integrity of each system becomes an important part for ensuring good quality software. However, that is not a one-time operation and needs to be performed over and over again. This is when automation comes into the picture.

Hence, Configuration Management, also sometimes referred to as Server Configuration Management (SCM) is an automation that ensures that the target system/server reaches its desirable state as per the configuration made upon it. This in turn is achieved with the help of Configuration Management Tools that have a variety of features to help with the provisioning and managing of the environment in a consistent manner.

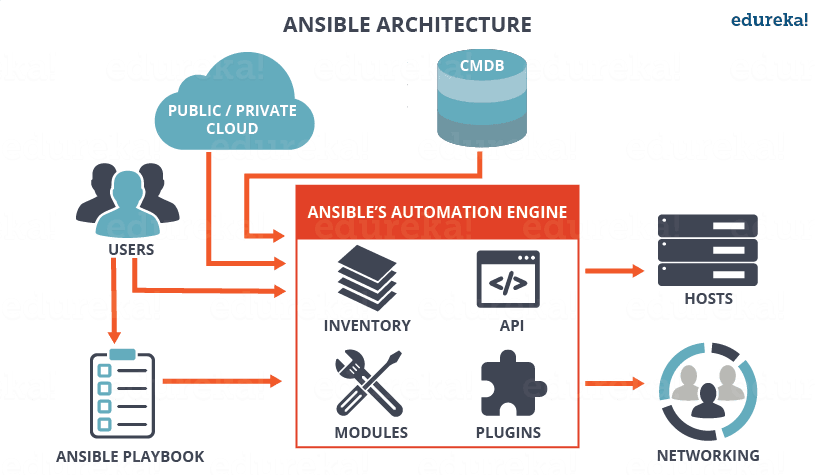
* Configuration management is defines the state of each system.

**What is Ansible?**

Open source automation platform which provides below use cases:

* **Configuration Management**
* **Application Deployment**
* Provisioning
* Continuous Delivery
* Security & compliance
* orchestration

**Ansible Architecture**



**How Ansible works?**

ANSIBLE

Control Node

HOST 1

[DEV SERVER]

HOST 2

[QA SERVER]

QA SERVER

HOST 3

[OD SERVER]

Inventory

Playbook

Modules

**Python program**

**[DEV SERVER] Yaml format**

**Host1.com**

**[QA SERVER]**

**Host2.com**

**How Ansible works?**

Ansible works by connecting to your nodes and pushing out small programs, called "Ansible modules" to them. These programs are written to be resource models of the desired state of the system. Ansible then executes these modules (over SSH by default), and removes them when finished.

**Ansible Components**

* **Inventory**

Ansible inventories are lists of hosts along with their IP addresses, servers, databases which needs to be managed. SSH is used for Unix based systems and WinRM for Windows based ones.

* **Modules**

These are programs executed on remote hosts through Ansible playbooks. They can be used to control services, system resources, and files etc on these hosts.

* **Plugins**

These represent external functionality that can add on to the existing functionalities of Ansible. There are Ansible provided plugins (ex: action, become, cache etc.) and we can create our own plugins too.

* **APIs:** APIs are used for interacting with the cloud

**Ansible Installation**

**Control node Managed node**

Any machine with Ansible installed. It’s a host/client without Ansible

Master Node Client node

Req: Python 2.7 or 3.5 Req: Python 2.6 or 3.5

Linux, Mac not windows SSH

SSH

$sudo dnf install https://dl.fedoraproject.org/pub/epel/epel-release-latest-8.noarch.rpm -y

$sudo dnf install ansible ……..

$ansible --version

**Ansible Features**

**Ansible is Agentless**

Typically, Ansible connects to the hosts using openssh or WinRM and runs tasks, often by pushing out small programs called “Ansible modules” to those hosts. These programs are used to put the system in desired state. Any modules that are pushed are removed when Ansible is finished with its tasks. So no agents and no additional customs security infrastructure needed.

**Easy to manage in version control**

Ansible playbooks and projects are plain text. They can treated like source code and placed in your existing version control system