Ansible

AUTOMATE APPS AND IT INFRASTRUCTURE

What is Ansible ?

- Ansible is an open source IT Configuration Management, Deployment & Orchestration tool. It aims to provide large productivity gains to a wide variety of automation challenges. This tool is very simple to use yet powerful enough to automate complex multi-tier IT application environments.

Ansible Terms:

- **Controller Machine**: The machine where Ansible is installed, responsible for running the provisioning on the servers you are managing.
- **Inventory**: An initialization file that contains information about the servers you are managing.
- **Playbook**: The entry point for Ansible provisioning, where the automation is defined through tasks using YAML format.
- **Task**: A block that defines a single procedure to be executed, e.g. Install a package.
- **Module**: A module typically abstracts a system task, like dealing with packages or creating and changing files. Ansible has a multitude of built-in modules, but you can also create custom ones.
- **Role**: A pre-defined way for organizing playbooks and other files in order to facilitate sharing and reusing portions of a provisioning.
- **Play**: A provisioning executed from start to finish is called a play. In simple words, execution of a playbook is called a play.
- **Facts**: Global variables containing information about the system, like network interfaces or operating system.
- **Handlers**: Used to trigger service status changes, like restarting or stopping a service.

Advantages of using Ansible

- **Simple:** Ansible uses a simple syntax written in YAML called ***playbooks***. YAML is a human-readable data serialization language. It is extraordinarily simple. So, no special coding skills are required and even people in your IT organization, who do not know what is Ansible can likely read a playbook and understand what is happening

- **Agentless:** Ansible is completely agentless. There are no agents/software or additional firewall ports that you need to install on the client systems or hosts which you want to automate

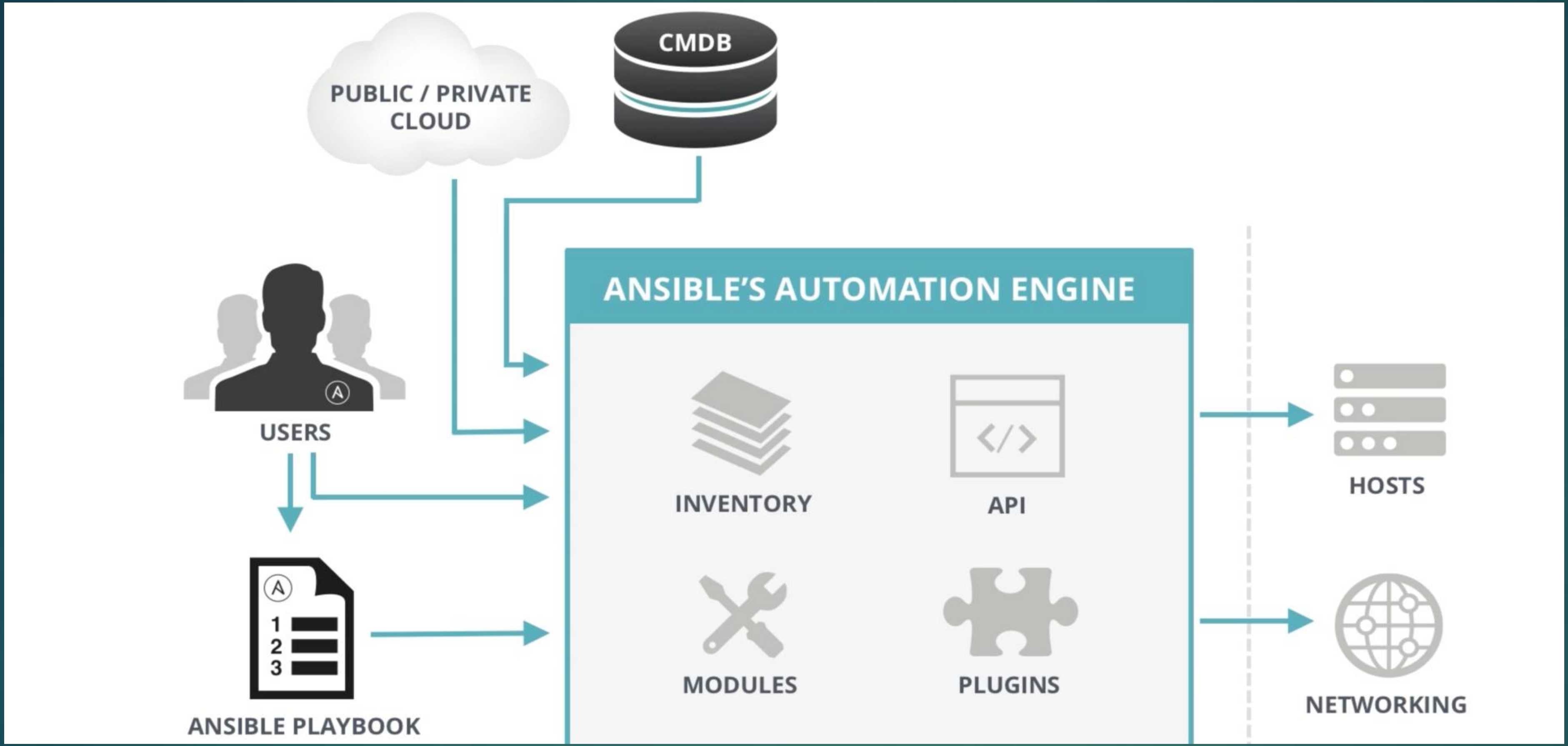
- **Powerful & Flexible:** Ansible has powerful features that can enable you to model even the most complex IT workflows. Ansible provides you with hundreds of modules to manage them. Together Ansible's capabilities allow you to orchestrate the entire application environment regardless of where it is deployed.

- **Efficient:** No extra software on your servers means more resources for your applications. Also, since Ansible modules work via JSON, Ansible is extensible with modules written in a programming language you already know. Ansible introduces modules as basic building blocks for your software. So, you can even customize it as per your needs. For e.g. If you have an existing message sending module which sends messages in plain-text, and you want to send images too, you can add image sending features on top of it.

What Ansible Can Do ?

- **Configuration Management:**
- It establishes and maintains consistency of the product performance by recording and updating detailed information which describes an enterprise’s hardware and software. Such information typically includes the 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 Tomcat 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 Tomcat 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 Tomcat. Run the playbook from your control machine & it will be installed on all your nodes.
- **Application Deployment:** When you define your application with Ansible, and manage the deployment with Ansible Tower, teams are able to effectively manage the entire application life cycle from development to production.
- **Orchestration**: Configurations alone don’t define your environment. You need to define how multiple configurations interact and ensure the disparate pieces can be managed as a whole. Out of complexity and chaos, Ansible brings order. Ansible provides Orchestration in the sense of aligning the business request with the applications, data, and infrastructure. It defines the policies and service levels through automated workflows, provisioning, and change management. This creates an application-aligned infrastructure that can be scaled up or down based on the needs of each application.

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)

Ansible Architecture

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.
- **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 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.

Ansible Architecture

- **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.

Writing Ansible Playbooks

- 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”.

Hosts And Users:

- For each play in a playbook, you get to choose which machines in your infrastructure to target and which remote user to complete the tasks.
- Generally the hosts are a list one or more groups or host patterns, separated by colons. The remote user is just the name of the user account.

Variables:

- Ansible uses variables which are defined previously to enable more flexibility in playbooks and roles. They can be used to loop through a set of given values, access various information like the host name of a system and replace certain strings in templates with specific values.
- Ansible already defines a rich set of variables, individual for each system. Whenever Ansible will run on a system, all facts and information about the system are gathered and set as variables.

Tasks:

- Tasks allow you to break up bits of configuration policy into smaller files. Task includes pull from other files. Tasks in Ansible go with pretty much the English meaning of it.
- E.g: Install <package_name>, update <software_name> etc.

Handlers:

- Handlers are just like regular tasks in an Ansible playbook, but are only run if the Task contains a notify directive and also indicates that it changed something. For example, if a config file is changed, then the task referencing the config file may notify a service restart handler.

Playbook Example

```
---
- hosts: webservers
  vars:
    http_port: 80
    max_clients: 200
  remote_user: root
  tasks:
  - name: ensure apache is at the latest version
    yum: name=httpd state=latest
  - name: write the apache config file
    template: src=/srv/httpd.j2 dest=/etc/httpd.conf
    notify:
    - restart apache
  - name: ensure apache is running (and enable it at boot)
    service: name=httpd state=started enabled=yes
  handlers:
    - name: restart apache
      service: name=httpd state=restarted
```