
ANSIBLE

➢ ANSIBLE:

- Ansible is an open-source software provisioning, configuration management, and application deployment tool.
- It runs on many Unix-like systems, and can configure both Unix-like systems as well as Microsoft Windows.
- Ansible was originally written by Michael DeHaan, the creator of the cobbler provisioning application.
- Ansible is simple to use for system administrator. Developers ease into using Ansible because it is built on Python.
- Ansible is included as part of the Fedora distribution of Linux, such as Red Hat, Centos, Debian, OEL…etc.
- Ansible's architecture is agentless. Work is pushed to remote hosts when Ansible executes.

WHY USE ANSIBLE?

- One of the most significant advantages of Ansible is that it is free to use by everyone.
- Its modularity regarding plugins, modules, inventories, and playbooks make Ansible the perfect companion to orchestrate large environments.
- Ansible is very lightweight and consistent, and no constraints regarding the operating system or underlying hardware are present.
- It is also very secure due to its agentless capabilities and due to the use of OpenSSH security features.

RED HAT ANSIBLE AUTOMATION:

- Red Hat Ansible Automation automates your entire IT infrastructure, using the expertise and knowledge already existing in your teams.
- Ansible Automation is used by thousands of organizations globally to help them automate IT tasks, such as configuration management, provisioning, workflow orchestration, application deployment, and life-cycle management. Ansible Automation is easy to adopt across the entire enterprise—from networks, servers, and security and compliance to cloud, infrastructure, and DevOps and continuous integration/continuous delivery (CI/CD).

ANSIBLE AUTOMATION IS SIMPLE:

- The language of Ansible Automation is easy to read. No special coding skills are needed. Your staff can start automating immediately.
- You do not need to install agents on servers, deploy expensive management appliances, or change your existing infrastructure.
- Ansible Automation uses standard communication mechanisms already in place on enterprise networks, such as secure shell (SSH) and Windows Remote Management (WinRM).
- No agents are required to be added to standard builds.

ANSIBLE AUTOMATION IS POWERFUL:

- Automate your Linux and Windows environments, your network, your clouds, and your applications. Ansible Automation brings these pieces together.
- Do more than just configuration management server by server. Orchestrate your entire IT landscape.
- Use Ansible Automation modules to automate existing tools, programs, and scripts under one umbrella.

RED HAT ANSIBLE TOWER:

- Part of the Red Hat Ansible Automation product family, Red Hat Ansible Tower is an enterprise framework for controlling, securing, and managing your automation with a user interface (UI) and a representational state transfer (RESTful) application programming interface (API). It integrates with your existing IT stack and brings it together in an automated fashion. Ansible Tower provides the control, knowledge, and delegation to help your organization automate your entire IT landscape and your critical business processes.

PUPPET ENTERPRICE (PE):

- Puppet Enterprise (PE) offers remote agentless capabilities and robust agent-based solutions to help you automate configuration management. Combine a model-driven approach with imperative task execution to manage hybrid infrastructure across its entire lifecycle. PE transforms infrastructure into code, so IT organizations can coordinate orchestrated application delivery and lifecycle management.

❖ IMPORTANT TERMS USED IN ANSIBLE:

ANSIBLE SERVER:

The machine where Ansible is installed and from which all tasks and playbooks will be ran

MODULE:

Basically, a module is a command or set of similar Ansible commands meant to be executed on the client-side

TASK:

A task is a section that consists of a single procedure to be completed

ROLE:

A way of organizing tasks and related files to be later called in a playbook

FACT:

Information fetched from the client system from the global variables with the gather-facts operation

INVENTORY:

File containing data about the ansible client servers. Defined in later examples as hosts file

PLAY:

Execution of a playbook

HANDLER:

Task which is called only if a notifier is present

NOTIFIER:

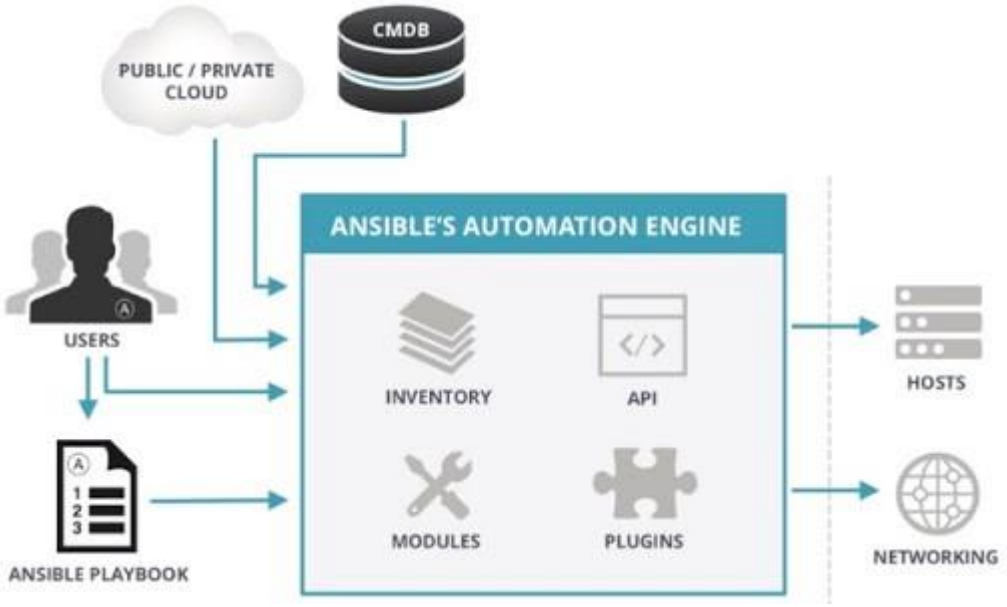
Section attributed to a task which calls a handler if the output is changed

TAG:

Name set to a task which can be used later on to issue just that specific task or group of tasks.

❖ ANSIBLE ARCHITECTURE:

- The Ansible orchestration engine interacts with a user who is writing the Ansible playbook to execute the Ansible orchestration and interact along with the services of private or public cloud and configuration management database.



INVENTORY:

- Inventory is lists of nodes or hosts having their IP addresses, databases, servers, etc. which are need to be managed.

API's:

- The Ansible API's works as the transport for the public or private cloud services.

MODULES:

- Ansible connected the nodes and spread out the Ansible modules programs. Ansible executes the modules and removed after finished. These modules can reside on any machine; no database or servers are required here. You can work with the chose text editor or a terminal or version control system to keep track of the changes in the content.

PLUGINS:

- Plugins is a piece of code that expends the core functionality of Ansible. There are many useful plugins, and you also can write your own.

PLAYBOOKS:

- Playbooks consist of your written code, and they are written in YAML format, which describes the tasks and executes through the Ansible. Also, you can launch the tasks synchronously and asynchronously with playbooks.

HOSTS:

- In the Ansible architecture, hosts are the node systems, which are automated by Ansible, and any machine such as RedHat, Linux, Windows, etc.

NETWORKING:

- Ansible is used to automate different networks, and it uses the simple, secure, and powerful agentless automation framework for IT operations and development. It uses a type of data model which separated from the Ansible automation engine that spans the different hardware quite easily.

CLOUD:

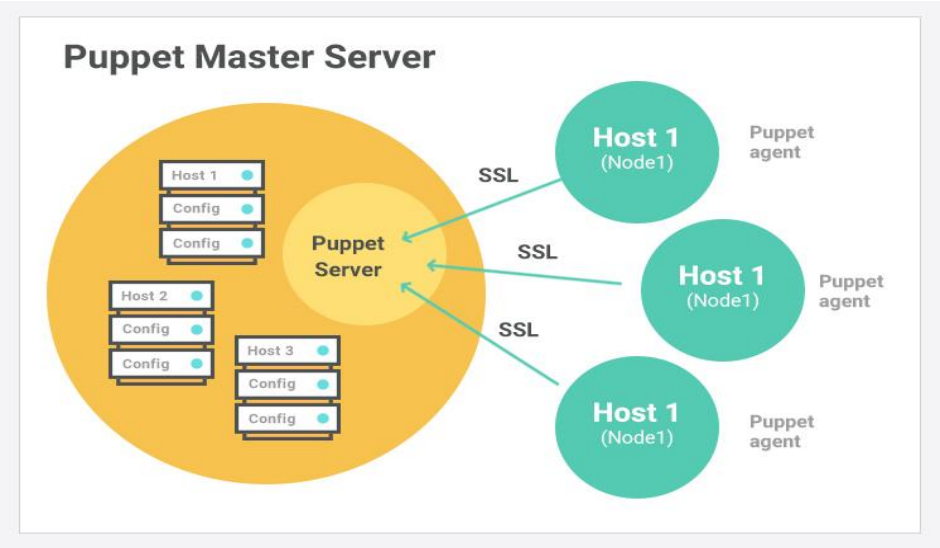
- A cloud is a network of remote servers on which you can store, manage, and process the data. These servers are hosted on the internet and storing the data remotely rather than the local server. It just launches the resources and instances on the cloud, connect them to the servers, and you have good knowledge of operating your tasks remotely.

CMDB:

- CMDB is a type of repository which acts as a data warehouse for the IT installations.

PUPPET ARCHITECTURE:

- Puppet is configured in an agent-server architecture, in which a primary server node controls configuration information for a fleet of managed agent nodes.



SERVER-AGENT COMMUNICATION FOLLOWS THIS PATTERN:

- An agent node sends facts to the primary server and requests a catalog.
- The primary server compiles and returns the node's catalog using the sources of information the primary server has access to.
- The agent applies the catalog to the node by checking each resource the catalog describes. If it finds resources that are not in their desired state, it makes the changes necessary to correct them. Or, in no-op mode, it assesses what changes would be needed to reconcile the catalog.
- The agent sends a report back to the primary server.
- primary servers and agents communicate by HTTPS using SSL certificates.

❖ TERRAFORM Vs ANSIBLE:

- Both of them are capable of provisioning the new cloud infrastructure and configuring the same with required application components.
- Both Terraform and Ansible are capable of executing remote commands on the virtual machine that is newly created. This means, both the tools are agentless.
- Terraform uses cloud provider APIs to create infrastructure and basic configuration tasks are achieved using SSH. The same goes with Ansible – it uses SSH to perform all the required configuration tasks.
- In general, both the tools are great in their own ways. They have an overlap of functions when it comes to infrastructure management.

	Terraform	Ansible
Type	Orchestration tool	Configuration management tool
Syntex	HCL	YAML
Language	Declarative	Procedural
Default approach	Mutable infrastructure	Immutable infrastructure
Lifecycle management	Does support	Doesn't support
Capabilities	Provisioning and configuring	Provisioning and configuring
Agentless	✓	✓
Masterless	✓	✓

❖ ORCHESTRATION VS. CONFIGURATION MANAGEMENT:

- Orchestration/provisioning is a process where we create the infrastructure – virtual machines, network components, databases, etc. Whereas, on the other hand, configuration management is a process of automating versioned software component installation, OS configuration tasks, network and firewall configuration, etc.

❖ DECLARATIVE VS. PROCEDURAL LANGUAGE

- Terraform uses uses HCL (Hashicorp Configuration Language) which is declarative in nature. It doesn't matter in which sequence the code is written. The code could also be dispersed in multiple files.
- Ansible uses YAML syntax to define the procedure to perform on the target infrastructure. Ansible YAML scripts are procedural in nature – meaning when you write the script, it will be executed from top to bottom.

❖ MUTABLE VS. IMMUTABLE INFRASTRUCTURE

- Mutability is an attribute associated with the underlying infrastructure that defines the way newer versions of applications and services are deployed. Deployment either takes place on existing infrastructure, or we can provision a completely new set of infrastructure for the same.
- The deployment practices typically determine whether the infrastructure is mutable or immutable. When newer versions of applications are released on the same infrastructure, it is called mutable. However, if the deployment happens on completely new infrastructure during releases, it is said to be immutable.

❖ STATE MANAGEMENT

- Terraform manages the entire lifecycle of the resources under its management. It maintains the mapping of infrastructure resources with the current configuration in state files. State management plays a very important role in Terraform.
- As opposed to this, Ansible does not support any lifecycle management. Since Ansible mainly deals with configuration management and considering it defaults to immutable infrastructure, any changes introduced in the configuration are executed automatically on the target resource.