

Introduction of Ansible

Ansible is a Configuration Management This is the process of configuring servers from one point of control.

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 systems.

- Handlers: Used to trigger service status changes, like restarting or stopping a service.

Ansible is a helpful tool that allows you to create groups of machines, describe how these machines should be configured or what actions should be taken on them. Ansible issues all commands from a central location to perform these tasks.

Advantages & Disadvantages:

Advantages:

1. Provisioning of Servers.

Setup of s/w's on servers can be done very easily from one point.

2. Reduction of usage of resources.

We require less time, money and human resources to configure Servers.

3. Handling Snowflake servers.

After a point of time all servers in the data center behave like snowflake servers i.e, they might be running on slightly different h/w and s/w configurations. Configuration Management tools can pick up this info in a simple setup file which can be reused to set up similar environments.

4. Disaster Recovery

In case of disaster recovery where we can loose an entire data center. We can recreate similar data centers with greater ease.

5. Idempotent

Ansible removes the modules once those are installed so expertly. It connects to the host machine, executes the instructions, and if it is successfully installed, then removes that code in which one was copied on the host machine.

Popular CM tools:

Ansible
Chef
Puppet
Saltstack

- Ansible is installed on one machine that is called a "Controller".
- All the remaining servers that we want to configure are called "managed nodes/hosts".
- Ansible uses "agentless" policy to configure the remote servers. ie we don't require any client side s/w of ansible to be present on the managed nodes.
- Ansible uses "push" methodology to push the configuration changes via password-less ssh.

Ansible Inventory host pattern

An Ansible pattern can refer to a single host, an IP address, an inventory group, a set of groups, or all hosts in your inventory. Patterns are highly flexible - you can exclude or require subsets of hosts, use wildcards or regular expressions, and more. Ansible executes on all inventory hosts included in the pattern.

Using patterns:

You use a pattern almost any time you execute an ad hoc command or a playbook. The pattern is the only element of an [ad hoc command](#) that has no flag. It is usually the second element:

```
ansible <pattern> -m <module_name> -a "<module options>"
```

For example:

```
ansible webserver -m service -a "name=httpd state=restarted"
```

In a playbook, the pattern is the content of the `hosts:` line for each play:

Pythonlife.in

```
- name:    <play_name>    hosts:
    <pattern>
```

For example:

```
- name: restart  webservershosts:
    webserver
```

Since you often want to run a command or playbook against multiple hosts at once, patterns often refer to inventory groups. Both the ad-hoc command and the playbook above will execute against all machines in the `webserver` group.

Common patterns:

This table lists common patterns for targeting inventory hosts and groups.

Description	Pattern(s)	Targets
All hosts	all (or *)	
One host	host1	
Multiple hosts	host1:host2 (or host1,host2)	
One group	webserver	
Multiple groups	webserver:dbserver	all hosts in webserver plus all hosts in dbserver

Excluding groups	webservers:!atlanta	all hosts in webbservers except those in atlanta
Intersection of groups	webservers:&staging	any hosts in webbservers that are also in staging

*Note: -You can use either a comma (,) or a colon (:) to separate a list of hosts. The comma is preferred when dealing with ranges and IPv6 addresses.

Once you know the basic patterns, you can combine them.

This example:

webservers:dbservers:&staging:!phoenix

targets all machines in the groups 'webservers' and 'dbservers' that are also in the group 'staging', except any machines in the group 'phoenix'.

Ansible Modules

Modules in ansible:-

1)command: This is used to execute linux commands on the managed nodes. This is the default module of Ansible.

2)shell: This is used to execute shell scripts or python scripts on the managed nodes. It is also used for running commands related to redirection and piping.

3)ping: Used to check if the remote servers are ping -able or not.

4)user: This is used to perform user administration on the remote servers like creating/deleting users, setting passwords, setting home directories etc.

5)copy: This is used to copy files and folders from the controller to the managed nodes.

- 6)fetch: This is used to copy files from the managed nodes to the controller.
- 7)file: Used to create/delete files or directories on the managed nodes.
- 8)apt: Used for s/w package management on the managed nodes like installing s/w's, deleting, upgrading etc. This works on ubuntu based machines.
- 9)yum: Similar to apt but it works on Centos, Redhat linux etc.
- 10)service: Used to start, stop and restart services on the managed nodes.
- 11)uri: Used to check if a url is reachable from the managed nodes.
- 12)git: Used to perform git version control on the remote managed nodes.
- 13)get_url: Used to download files from remote servers works like linux command wget.
- 14)stat: Captures info about files and folders present on the managed nodes.
- 15)debug: This is the print statement of ansible.
- 16)include: Used to call child playbooks from the level of a parent playbook.
- 17)replace: Used to change specific sections of the file.
- 18)pause: Used to pause the playbook execution for a specific period.
- 19)docker_container: Used to handle docker containers on the managed nodes.
- 20)docker_image: Used to handle docker images on the managed nodes.

Host Patterns

Ansible perform remote configuration in 3 ways

- 1) Ad-hoc commands
- 2) Playbooks
- 3) Roles

Ad-hoc Commands

Syntax of Ad-hoc commands:

- ```
ansible -i inventory mongodb --become -e
ansible_username=centos -e ansible_password=DevOps321 -m
ansible.builtin.service -a "name=nginx state=started"
```

```
ansible -i inventory mongodb --become -e
ansible_username=centos -e ansible_password=DevOps321 -m
m ansible.builtin.yum -a "name=nginx state=installed"
```

## **Command Module:**

Ansible command to see the memory information of all managed nodes.

```
ansible all -i /etc/ansible/hosts -m command -a 'free m'
```

`/etc/ansible/hosts` is the default inventory file and when using it we need not give `-i`. 

```
ansible all -m command -a 'free m'
```

Command module is the default module of ansible and we need not use `-m` option when working on command module

```
ansible all -a 'free m'
```

## **Shell Module:**

Ansible commands to download the docker script and execute it to install docker.

```
ansible all -m shell -a 'curl -fsSL https://get.docker.com -
o get-docker.sh' ansible all -m shell -a 'sh get-docker.sh'
```

**Pythonlife.in**



Ansible command to capture memory statics into a file called as file1

```
ansible all -m shell -a 'free -m > file1'
```

## Playbooks

### Ansible Playbooks

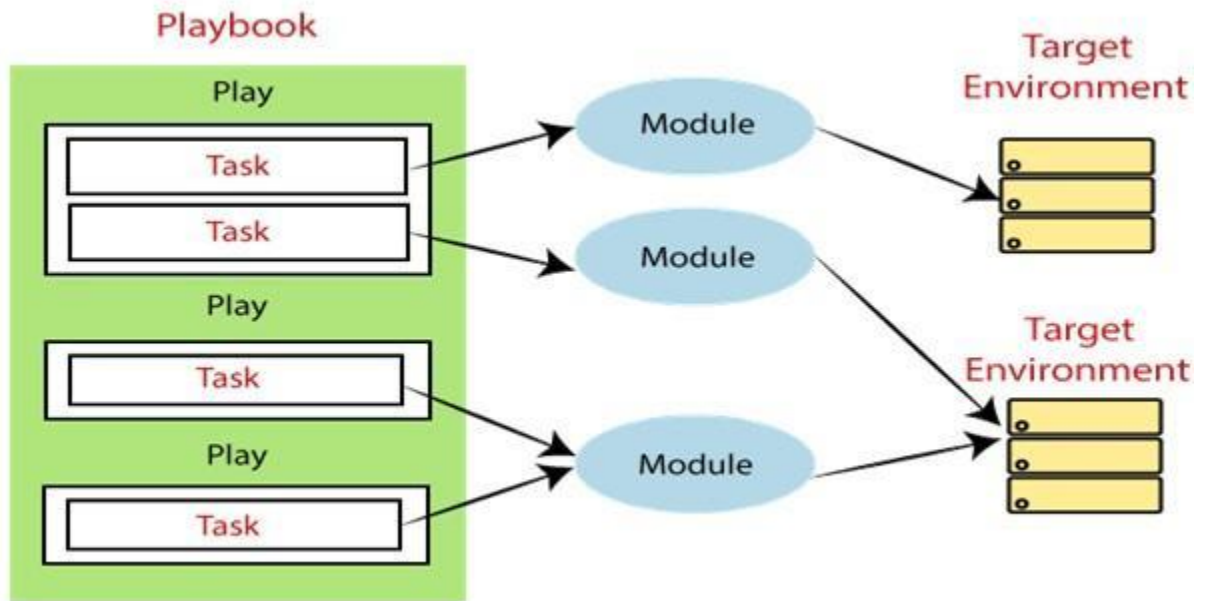
Ad Hoc commands become difficult to handle when working on complex configurations of s/w applications. Each ad hoc command can work only on one module and one set of arguments. In such cases we can use Ansible playbooks which support greater reusability. Playbooks are created using yaml and each playbook is a combination of multiple plays. A play contains info about what module has to be executed. These plays are designed to work on a single host or a group of hosts or all the hosts.

Playbooks are the files where the Ansible code is written. Playbooks are written in YAML format. YAML means "Yet Another Markup Language," so there is not much syntax needed. Playbooks are one of the core features of Ansible and tell Ansible what to execute, and it is used in complex scenarios. They offer increased flexibility. Playbooks contain the steps which the user wants to execute on a particular machine. And playbooks are run sequentially. Playbooks are the building blocks for all the use cases of Ansible.

Ansible playbooks tend to be more configuration language than a programming language. Through a playbook, you can designate specific roles to some of the hosts and other roles to other hosts. By doing this, you can orchestrate multiple servers in very different scenarios, all in one playbook.

### Playbook Structure

Each playbook is a collection of one or more plays. Playbooks are structured by using Plays. There can be more than one play inside a playbook.



## Yaml

YAML means "Yet Another Markup Language," so there is not much syntax needed. There are different YAML editors, but prefer to use a simple editor such as notepad++. First, open the notepad++ and copy-paste the below YAML and change the language to YAML (Language → YAML).

A YAML starts with --- (3 hyphens) always.

### YAML Tags:

Here are some YAML tags are given below, such as:

| Tags        | Explanation                                     |
|-------------|-------------------------------------------------|
| <b>Name</b> | It specifies the name of the Ansible Playbooks. |

|              |                                                                                                                                                                                                                                                                                                                                                                                                                      |
|--------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Hosts</b> | <p>It specifies the lists of the hosts against which you want to run the task. And the host's Tag is mandatory. It tells Ansible on which hosts to run the listed tasks. These tasks can be run on the same machine or the remote machine. One can run the tasks on the multiple machines, and the host's tag can have a group of host's entries as well.</p>                                                        |
| <b>Vars</b>  | <p>Vars tag defines the variables which you can use in your playbook. Its usage is similar to the variables in any programming language.</p>                                                                                                                                                                                                                                                                         |
| <b>Tasks</b> | <p>Tasks are the lists of the actions which need to be performed in the playbooks. All the playbooks should contain the tasks to be executed. A task field includes the name of the task. It is not mandatory but useful for debugging the playbook. Internally each task links to a piece of code called a module. A module should be executed, and arguments that are required for the module you want to run.</p> |

## Variables

### Ansible Variables

In playbooks, the variable is very similar to using the variables in a programming language. It helps you to assign a value to a variable and use it anywhere in the playbook. You can put the conditions around the value of the variables and use them in the playbook accordingly.

Variables are categorized into 3 types.

- 1) Global scope variables
- 2) Play scope variables
- 3) Host Scope variables

**To Install NGINX the following is the syntax:**

- name: install and run nginx

hosts: mongodb

become: yes

tasks:

- name: INSTALL NGINX

ansible.builtin.yum:

name: nginx

state: installed

- name: start nginx

ansible.builtin.service:

name: nginx

state: started

**To Print hello world following is the syntax:**

- name: variable in ansible

hosts: mongodb

tasks:

- name: print hello world

ansible.builtin.debug:

**Pythonlife.in**

```
msg: "hello ,i am learning ansible"
```

## **Implementing variables topic in ansible:**

- name: variable in ansible

```
hosts: mongodb
```

```
vars:
```

```
COURSE: devops
```

```
TRAINER: ragu
```

```
DURATION: 15hrs
```

```
tasks:
```

- name: print hello world

```
ansible.builtin.debug:
```

```
msg: "hello, i am learning ansible"
```

- name: print variables

```
ansible.builtin.debug:
```

```
msg: "hello, i am learning {{COURSE}}, trainer is {{TRAINER}}, duration is {{DURATION}}"
```

## **Prompting the variables in ansible topic:**

- name: variables from prompt

**Pythonlife.in**

```
hosts: localhost

vars_prompt:
- name: USERNAME
 prompt: pls enter your username
 private: false
- name: PASSWORD
 prompt: pls enter your password
 private: true

tasks:
- name: print variable values
 ansible.builtin.debug:
 msg: "username: {{USERNAME}}, password: {{PASSWORD}}"
```

## Handlers

- 1) Handlers are modules that are executed if some other module is executed successfully and it has made some changes.
- 2) Handlers are only executed after all the modules in the tasks section are executed.
- 3) Handlers are executed in the order that they are mentioned in the handler's section and not in the order that they are called in the tasks section.
- 4) Even if a handler is called multiple times in the tasks section it will be executed only once.

```
vim playbook15.yml
```

## Error Handling

Whenever a module in ansible playbook fails the execution of the playbook stops there, if we know that a specific module can fail and still we want to continue the execution of the playbook we can use error handling.

The module that might fail should be given in the "block" section, if it fails the control comes to the "rescue" section "always" section is executed every time.

## Loops in Ansible

Loops can be implemented in Ansible using with\_items and with\_sequence. Ansible playbook to install multiple s/w applications.

```
- name: loops example

hosts: localhost

tasks:

- name: print the names

 ansible.builtin.debug:

 msg: "hello {{item}}"

 loop:

 - ragu

 - siva

 - jhon
```

```
- viya
```

```
- name: Install common packages

action: "{{ ansible_pkg_mgr }}"

with_items:

 - { name: bash-completion, state: present }

 - { name: libselinux-python, state: present }

 - { name: logwatch, state: present }

 - { name: sssd, state: present }
```

## Conditions

When conditions

This is "if" conditions and it helps us to execute modules based on a specific Condition.

Create a file based on a condition

```
- name: simple condition
hosts: localhost
vars:
 NAME: DevOps
tasks:
- name: run this if name is DevOps
 ansible.builtin.debug:
 msg: "Hello.... {{NAME}}"
 when: NAME == "DevOps"
```



## Ansible Vault

This is a feature of ansible which allows us to protect the playbooks via a password. Playbooks created using vault can be viewed, edited or executed only if we know the password.

- 1) To create a vault `ansible-vault create playbook_name.yml`
- 2) To view the content of a vault `ansible-vault view playbook_name.yml`
- 3) To edit the content of a vault `ansible-vault edit playbook_name.yml`
- 4) To convert an ordinary playbook into a vault `ansible-vault encrypt playbook_name.yml`
- 5) To convert a vault playbook into an ordinary `ansible-vault decrypt playbook_name.yml`
- 6) To reset the password of a vault `ansible-vault rekey playbook_name.yml`

### **module:**

This is used to call child playbooks from the level of a parent playbook

## Roles in Ansible

Roles provide greater reusability than playbooks. Generally, roles are used to configure s/w applications. Everything necessary to configure a s/w application should be present with the folder structure of a role. This aids in easy understanding and maintenance of CM activities.

Roles should be created in `/etc/ansible/roles` folder

To create roles in some other locations.

```
sudo vim /etc/ansible/ansible.config
```

Search for `roles_path` and give the path of the directory where we want to create the role and uncomment it.

## Folder structure of roles

README.MD: This is a simple text file that is used to store info about the role in plain English.

defaults: This stores info about the application that we are configuring and it also stores variables of lesser priority.

files: All the static files that are required for configuring a s/w application are stored here.

meta: Data about the data is called as metadata and this is used to store info about the roles like when it was created, who created it, what versions it supports etc.

handlers: handlers are modules that are executed when some other module is successful and it has made some changes, all such handlers are stored in this folder.

tasks: The actual configuration management activity that has to be performed on the remote servers is stored in this folder.

templates: This is used to store dynamic configuration files.

tests: All the modules that are used to check if the remote configurations are successful or not are stored in this folder.

vars: This is used to store all the variables that are required for configuring a specific s/w application. These variables have higher priority than the variables in the defaults folder.

# Ansible Galaxy

Ansible Galaxy is a galaxy website where users can share roles and a command-line tool for installing, creating, and managing roles.

Ansible Galaxy gives greater visibility to one of Ansible most exciting features, such as application installation or reusable roles for server configuration. Lots of people share roles in the Ansible Galaxy.

Ansible roles consist of many playbooks, which is a way to group multiple tasks into one container to do the automation in a very effective manner with clean, directory structures.

## Ansible Galaxy Commands

Here are some helpful Ansible Galaxy commands, such as:

- To display the list of installed roles, with version numbers.

```
ansible-galaxy list
```

- To remove an installed role.

```
ansible-galaxy remove [role]
```

- To create a role template suitable for submission to Ansible Galaxy.

```
ansible-galaxy init
```

The `ansible-galaxy-collection` command implements the following commands. Some commands are the same as used with `ansible-galaxy`, such as:

init: It creates a basic collection Skeleton based on the default template included with Ansible or your own template.







build: It creates a collection artifact that can be uploaded to the galaxy or your own repository.

publish: It publishes a built connection artifact to the galaxy.

**Pythonlife.in**

install: It installs one or more connections.

## Difference Between Tools

| PRODUCTS                                                                                             | Best Suited For                              | Functions Or Key Plugin                                     | Tool Cost                                                                            | Few Companies Using These Tools                                                 |
|------------------------------------------------------------------------------------------------------|----------------------------------------------|-------------------------------------------------------------|--------------------------------------------------------------------------------------|---------------------------------------------------------------------------------|
|  <b>ANSIBLE</b>     | Cloud Provisioning<br>Application Deployment | No Additional Agents<br>Required                            | Ansible Tower gives free<br>license upto 10 Nodes                                    | NASA, ViaSat,<br>Capital One                                                    |
|  <b>puppet</b>      | Configuration management                     | Simplicity                                                  | A standard subscription<br>would cost approx \$39k<br>per year for 100 virtual nodes | Google, Red Hat,<br>New York Stock Exchange,<br>Harvard & Stanford Universities |
|  <b>Jenkins</b>     | Continuous Integration process               | Git, Amazon EC2,<br>Maven 2 project                         | Basic setup is between<br>\$72 to \$163/month                                        | Linkedin, Cppgemini                                                             |
|  <b>CHEF</b>        | Configuration management                     | Configuration policies are<br>Flexible, readable & testable | Free                                                                                 | Facebook, Hewlett Packard<br>Enterprise, Firefox                                |
|  <b>docker</b>      | Container Technology                         | Easy & Faster Configuration,<br>Increase productivity       | Starting price is \$750/node<br>annually                                             | Paypal, VISA, Indiana<br>University, MetLife                                    |
|  <b>Consul</b>      | Service Discovery Tool                       | Health Checking, Secure<br>service communication            | Customized packages                                                                  | Digital ocean, Slack,<br>SendGrid, Wildcard                                     |
|  <b>etcd</b>        | Distributed key or value store               | Service discovery                                           | Multiple subscription models                                                         | Salesforce, Pokemon Go,<br>Niantic                                              |
|  <b>New Relic</b> | Application & Server<br>Monitoring Tool      | Easy UI, No Installation,<br>Does Capacity analysis         | \$199 monthly & \$149 per<br>month per host                                          | TopGolf, Alkarni, Telenor group                                                 |
|  <b>sensu</b>     | Monitoring Tool                              | Instant Alerts, Auto-remediation &<br>Scheduling            | \$899 per month                                                                      | SendGrid, Opower, Zinc                                                          |



