**Ansible**

It is an open-source tool used for configuration management, application deployment, along with infrastructure orchestration

* It is a push-based configuration management tool
* It does not require any dedicated agent running on the target host machines.
* It has a requirement for host machines is python installed on them and a proper SSH connection between the controller and the host machines has to be established

**Installation:**

Redhat:

* sudo yum update
* sudo yum install ansible

Ubuntu:

* sudo apt add-repository ppa:ansible/ansible
* sudo apt update
* sudo apt install ansible

**Architecture:**

**Modules**: Ansible works by connecting to your nodes and pushing out scripts called “**Ansible modules**” to them, Ansible then executes these modules (over SSH by default), and removes them when finished.

**Plugins**: While modules execute on the target system, plugins execute on the control node. Plugins offer options and extensions for the core features of Ansible

**Controller Machine:** Controller is a Machine where Ansible is installed

**Inventory:**

* Inventory is the place where we put information about all our managed machine i.e., host machines
  + Static Inventory
  + Dynamic Inventory

**Static Inventory:**

* It is a file which contains the IP and configuration of target host machines on which we want execute our playbooks
* Default location of host file is /etc/ansible/hosts

**Syntax:**

* [group\_name]
* <ip\_address> ansible\_user=<user\_name> ansible\_become\_user=root

**AD-Hoc Commands:**

ansible <hosts> -m <module> -a <arguments>

**Example:**

ansible all -m ping

ansible <group\_name> -m shell -a "mkdir test"

**Ansible Playbooks:**

**Playbook**: Playbooks are list of tasks written in <.yaml> format that gets executed on host machines

* Can contain n number of plays
* Each play is designated to run n number tasks
* Each task is designated to execute a single module (i.e., only one module per task)

To Execute Playbook: **Ansible-playbook <file\_name>.yaml**

**Simple Playbook:** to install git

- name: simple-playbook

hosts: all

tasks:

- name: Install Git

become: true

yum:

name: git

state: present

update\_cache: yes

**become:**

* **true** will execute the task as root i.e sudo

**State:**

* **Present**: ensure that a desired package is installed
* **latest**: update the specified package if it's not of the latest available version.
* **absent**: remove the specified package
* **build-dep:** install any dependency packages along with the desired package

**Playbook with 🡪 Multiple Tasks:**

- name: simple-playbook

hosts: all

tasks:

- name: Install Git

become: true

dnf:

name: git

state: present

update\_cache: yes

- name: Install wget

become: true

dnf:

name: wget

state: present

update\_cache: yes

**update\_cache:**

* **yes:** tells Ansible’s apt module to refresh the caches before applying whatever change is necessary (i.e., apt-get update)

**Installing Multiple Packages within single Task**

- name: simple-playbook

hosts: all

tasks:

- name: Multi Install

become: true

dnf:

name: ['git', 'wget', 'unzip']

state: present

update\_cache: yes

**Ansible Facts:**

* Ansible facts are data gathered about host machines and returned back to the controller.
* Ansible facts are stored in JSON format and setup module is used to gather all the facts

- name: simple-playbook

hosts: all

tasks:

- name: Multi Install

become: true

apt:

name: ['git', 'wget', 'unzip']

state: present

update\_cache: yes

when:

- ansible\_distribution == "Ubuntu"

- ansible\_pkg\_mgr == "apt"

- name: Multi Install

become: true

dnf:

name: ['git', 'wget', 'unzip']

state: present

update\_cache: yes

when:

- ansible\_distribution == "RedHat"

- ansible\_pkg\_mgr == "dnf"

**ansible all -m setup** 🡪 To gather all Ansible facts

**Ansible Register and Debugging:**

* Ansible register is a way to capture the output from task execution and store it in a variable
* Ansible run a task only if previous task is successfully changed using register module
* Register module will save the output of a task to a variable in JSON format only if the task executes successfully
* We can register the output of one task and on another task, we can check whether variable has information or empty

- name: simple-playbook

hosts: frontend

tasks:

- name: Multi Install

become: true

dnf:

name: ['git', 'wget', 'unzip']

state: present

update\_cache: yes

register: test

- name: Print Debug Message

debug:

var: test

**To run an Ansible Task only if the previous task has made some changes to host machines**

- name: simple-playbook

hosts: frontend

tasks:

- name: Install Git

become: true

dnf:

name: git

state: present

update\_cache: yes

register: result

- name: Install wget

become: true

dnf:

name: wget

state: present

update\_cache: yes

when: result is changed

**Ansible Tags:**

* using tags, we can run specific task or play in a playbook

- name: simple-playbook

hosts: backend

tasks:

- name: Install Git

become: true

dnf:

name: git

state: present

update\_cache: yes

tags: git

- name: Install wget

become: true

dnf:

name: wget

state: present

update\_cache: yes

tags: wget

- name: Install unzip

become: true

dnf:

name: unzip

state: present

update\_cache: yes

tags: unzip

**To run only task with the tag**

* **ansible-playbook <playbook\_name> --tags="<tag\_name>"**
* **ansible-playbook <playbook\_name> --tags="<tag\_name1>, <tag\_name2>"**
* **ansible-playbook <playbook\_name>.yaml --tags="unzip "**
* **ansible-playbook <playbook\_name>.yaml --tags="git,unzip"**

**To skip a tagged play / task**

* **ansible-playbook <playbook\_name> --skip-tags="<tag\_name1>, <tag\_name2>"**
* **ansible-playbook <playbook\_name>.yaml --skip-tags="git"**
* **ansible-playbook <playbook\_name>.yaml --skip-tags="git,unzip"**

**Dynamic Inventory:**

* With scalable cloud infrastructure it is difficult to maintain a static inventory file as we might miss those instances that are newly created or created by auto-scaling.
* Ansible provides a way to capture these new host machines with Dynamic Inventory concept using plugins

**Pre-requisites**

1. Install Python and Boto3

**sudo yum install python36**

**python3 -m pip install --user --upgrade pip**

**pip install boto3**

2. Install AWS CLI and configure your AWS access key and secret key

plugin: aws\_ec2

regions:

ap-southeast-1

keyed\_groups:

- key: tags

prefix: tag

- key: instance\_types

prefix: instance\_type

**Configuration changes in ansible.cfg**

1. inventory = /etc/ansible/dynamic/aws\_ec2.yaml // path of plugin yaml file

2. enable\_plgins = aws\_ec2

**Commands:**

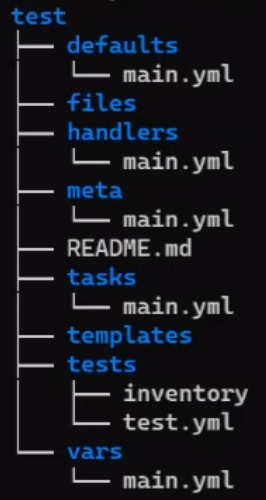
* **ansible-inventory –graph** 🡪 To check all the host machine in the inventory
* **ansible <tag> -m ping**

**Ansible Roles:**

* Ansible Roles are a way of breaking down a playbook into multiple files. Therefore, simplifying writing complex playbooks and making them easier to reuse.
* Ansible provides a feature called **Ansible Galaxy** that helps you create roles.

Default Path: **/etc/ansible/roles/** [Create the folder if not present]

**ansible-galaxy init <role\_name>** → To create a Role

****

<role\_name>

├── defaults

│   └── main.yml

├── files

├── handlers

│   └── main.yml

├── meta

│   └── main.yml

├── README.md

├── tasks

│   └── main.yml

├── templates

├── tests

│   ├── inventory

│   └── test.yml

└── vars

└── main.yml

**Defaults** — Contains the default variables that are going to be used by this role. It has the least precedence in Ansible Roles

**Vars** — This directory consists of other variables that are going to be used by the role.

**Handlers** — The handlers directory is used for storing Ansible handlers. Handlers are the tasks which are executed only if they are invoked by another task

**Meta** — Defines metadata for this role. Basically, it contains files that establish role dependencies.

**Tasks** — Contains the main list of tasks that are to be executed by the role. It contains the main.yaml file on which we define our tasks

**Files** — Contains static files that can be deployed to the host machines by this role.

**templates** – Similar to files templates are used to copy files to host machines but with dynamic values. This uses jinja2 format templating.

**vi installpkg/tasks/main.yaml**

- name: Install Package

become: true

yum:

name: "{{ application }}"

state: "{{ state }}"

update\_cache: yes

**Vi installpkg/vars/main.yaml**

application: git

state: present

**vi execute-role.yaml**

-name: Execute Role

hosts: backend

become: true

roles:

- installpkg

**ansible-playbook execute-role.yaml 🡪**  to execute playbook

[This command won’t work for root user]

Docker Role: <https://github.com/adhig93/docker-install-ubuntu-role>

Tomcat Role: <https://github.com/adhig93/tomcat-role>

**Assignment:**

**Ansible Vault:**

It allows users to **encrypt** values and data structures within Ansible projects. A utility called ansible-vault secures confidential data, called *secrets*, by **encrypting it on disk**.

* **ansible-vault create vault.yml**
* **ansible-vault create --vault-id password@prompt vault.yml** 🡪 create with vault id
* **ansible-vault decrypt secure.yml**
* **ansible-vault edit secure.yml**

**Ansible Tower**

It is a web-based solution that makes Ansible even more easy to use for IT teams of all kinds. It's designed to be the hub for all of your automation tasks. managed by RHEAL

Prometheus monitoring for Kubernetes Cluster and Grafana visualization:

<https://www.youtube.com/watch?v=CmPdyvgmw-A&t=1710s>