**Rampup topics :**

* Understand core components of Ansible
  + Inventories
  + Modules
  + Variables
  + Facts
  + Plays
  + Playbooks
  + Configuration files
* Install and configure an Ansible control node
  + Install required packages
  + Create a static host inventory file
  + Create a configuration file
* Configure Ansible managed nodes
  + Create and distribute SSH keys to managed nodes
  + Configure privilege escalation on managed nodes
  + Validate a working configuration using ad-hoc Ansible commands
* Create simple shell scripts that run ad hoc Ansible commands
* Use both static and dynamic inventories to define groups of hosts
* Utilize an existing dynamic inventory script
* Create Ansible plays and playbooks
  + Know how to work with commonly used Ansible modules
  + Use variables to retrieve the results of running commands
  + Use conditionals to control play execution
  + Configure error handling
  + Create playbooks to configure systems to a specified state
* Use Ansible modules for system administration tasks that work with:
  + Software packages and repositories
  + Services
  + Firewall rules
  + File systems
  + Storage devices
  + File content
  + Archiving
  + Scheduled tasks
  + Security
  + Users and groups
* Create and use templates to create customized configuration files
* Work with Ansible variables and facts
* Create and work with roles
* Download roles from an Ansible Galaxy and use them
* Manage parallelism
* Use Ansible Vault in playbooks to protect sensitive data
* Use provided documentation to look up specific information about Ansible modules and commands

**eAnsible 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.

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

**Installation** :

* Add users : Useradd ansible , passwd ansible
* Edit sshd\_config : **/etc/ssh/sshd\_config - >** **sswordAuthentication and PermitRootLogin**
* **Systemctl restart sshd**
* **Cd /etc/sudoers** : add ansible ALL=(ALL) NOPASSWD:ALL
* (root) Add in inventory : vi /etc/ansible/hosts ->webservers -> ip of remote machine
* Su –ansbile
* Generate ssh key : ssh-keygen
* Copy the ssh key to target machine : ssh-copy-id 172.26.41.23
* Login as ansibel user : ssh [ansible@172.26.41.23](mailto:ansible@172.26.41.23)
* Sudo yum install wget –y
* **wget http://dl.fedoraproject.org/pub/epel/epel-release-latest-7.noarch.rpm**
* **sudo rpm -ivh epel-release-latest-7.noarch.rpm**
* **sudo yum install ansible -y**
* **ansible --version**
* to test connectivity between servers : ansible webservers -m ping
* list hosts in inventory file : ansible webservers --list-hosts

**modules**

1. **setup module :**

* information about the network or hardware or OS version or memory related information the setup module will help to gather the same about the target machines.
* ansible webservers -m setup

1. **command module**

* executes a specific command on the target machine and gives the output.
* ansible webservers -m command -a 'hostname'

1. **shell module**

* executes a specific command on the target machine and gives the output.with operators
* ansible webservers -m shell -a 'ls -l>temp.txt'
* ansible webservers -m command -a 'cat temp.txt'

1. **user module**

* cat /etc/passwd | grep Anamika
* create a user : ansible webservers -m user -a 'name=anamika password=anamika' –become
* delete a user: ansible webservers -m user -a 'name=anamika state=absent' –become

1. **file module**

* create file : ansible webservers -m file -a 'dest=/home/ansible/anamika.txt state=touch mode=600 owner=ansible group=ansible'
* delete a file : ansible webservers -m file -a "dest=/home/ansible/anamika.txt state=absent"
* create directory : ansible webservers -m file -a "dest=/home/ansible/newdirectory state=directory mode=755"
* delete a directory : ansible webservers -m file -a "dest=/home/ansible/newdirectory state=absent"

1. **copy module** :used for copying files to multiple target machines.

* copy a file: ansible webservers -m copy -a "src=anamika.txt dest=/home/ansible/newdirectory/anamika.txt"

1. **managing softwares:**

* install git : ansible webservers -m yum -a "name=git state=present" –become
* httpd : ansible webservers -m yum -a "name=httpd state=present" –become
* maven present or not : ansible webservers -m yum -a "name=maven state=absent" –become

1. **managing services**:

* start a service

ansible webservers -m service -a "name=httpd state=started" --become

* stop :

ansible webservers -m service -a "name=httpd state=stopped " --become

* restart:

ansible webservers -m service -a "name=httpd state=restarted " --become

**ansible command line tools :**

**ansible , ansible-config , ansible-console , ansible-doc, ansible-galaxy, ansible-inventory , ansible-playbook , ansible-pull , ansible-vault**

1. **ansible : ansible <host-pattern> <options>**

**options :**

**--ask-su-pass :** ask for su password

**--ask-sudo-pass :** ask for sudo password

**--ask-vault-pass :** ask for vault password

**--become-user :** run operations as this user

**--become-method :** privilege escalation method to use (default=sudo),

**--list-hosts :** outputs a list of matching hosts; does not execute anything else

**--playbook-dir<basedir> :** this as a subsitute playbook directory.This sets the relative path for many features including roles/ group\_vars/ etc.

**--vault-id** the vault identity to use

**--vault-password-file** vault password file

**--ask-become-pass** ask for privilege escalation password

**--check** don’t make any changes; instead, try to predict some of the changes that may occur

**--diff w**hen changing (small) files and templates, show the differences in those files;

**--su\_user** run operations with su as this user

**--timeout** override the connection timeout in seconds (default=10)

**--become**run operations with become

**-- forks** specify number of parallel processes to use (default=5)

**-- inventory** specify inventory host path or comma separated host list.

**--tree** log output to this directory

**--user<remote user>** connect as this user (default=None)

**--verbose** verbose mode (-vvv for more, -vvvv to enable connection debugging)

1. **ansible-config <sctions>**

ansible playbooks :

1. Every playbook starts with 3 hyphens ‘—‘
2. **Host section** – Defines the target machines on which the playbook should run. This is based on the Ansible inventory file.
3. **Variable section** – This is optional.
4. **Tasks section** –lists out all the tasks that should be executed on the target machine. It specifies the use of Modules. Every task has a name which is a small description of what the task will do and will be listed while the playbook is run.
5. **To run any playbook use the following command**

**ansible-playbook <playbook.yml>**

1. **To check the playbook for syntax errors**

**ansible-playbook <playbook.yml> --syntax-check**

1. **To view hosts list**

**ansible-playbook <playbook.yml> --list-hosts**

**playbook example:**

- hosts: webservers

become: true

tasks:

1. create the file on the target machines

- name: Create a file

file: path=/home/ansible/niranjan.txt state=touch

2. Create a directory with the mode as 775

- name: Create directory

file: path=/home/ansible/niranjan state=directory mode=775 owner=ansible group=ansible

3. Create multiple directories.

- name: Create multiple directories

file: path={{item}} state=directory

with\_items:

- '/home/ansible/vn1'

- '/home/ansible/vn2'

- '/home/ansible/vn3'

4. Create a user.

- name: Create User

user: name=niranjan password=niranjan groups=ansible shell=/bin/bash

5. Remove user. Removing a user is very easy and it will need the **state** to be set to **absent**.

- name: Remove User

user:

name=niranjan state=absent remove=yes force=yes

6. Copy content to a file using the copy module.

- name: Copy content to file

copy: content="Hello World Niranjan \n" dest=/home/ansible/niranjan.txt

copy: src=/home/ansible/niranjan.txt dest=/tmp/niranjan.txt

1. Replace all instances of a string

- name: Replace example

replace:

path: /home/ansible/niranjan.txt

regexp: 'hello'

replace: "world"

1. ZIP files and Folders

playbook will zip the file niranjan.txt to niranjan.zip file

- name: Ansible zip file example

archive:

path: /home/ansible/niranjan.txt

dest: /home/ansible/niranjan.zip

format: zip (playbook will zip the file niranjan.txt to niranjan.zip file)

9. playbook will zip multiple files to niranjan.zip file.

- name: Ansible zip multiple files example

archive:

path:

- /home/ansible/niranjan1.txt

- /home/ansible/niranjan2.txt

dest: /home/ansible/niranjan.zip

format: zip

10. zip all files in the /home/ansible directory.

- name: Ansible zip directory example

archive:

path:

- /home/ansible

dest: /home/ansible/niranjan.zip

format: zip

11. display date n time

- name: Date and Time Example in Ansible

debug:

var=ansible\_date\_time.time

var=ansible\_date\_time.date

12. variables :

hosts: all

vars:

name:

-anamika

-aman

tasks:

- name: Ansible array variable Example

debug:

msg: "{{ name[1] }}"

**ansible\_connection :** Connection type to the host. This can be the name of any of ansible’s connection plugins..

1. Connection types : ssh , non ssh

2. Ssh types : smart (default) ,ssh ,

**For all:**

**ansible\_host :** The name of the host to connect to,

**ansible\_port :** The ssh port number, if not 22

**ansible\_user :** The default ssh user name to use.

**SSH connection:**

* **ansible\_ssh\_pass :** ssh password to use
* **ansible\_ssh\_private\_key\_file** Private key file used by ssh. Useful if using multiple keys and you don’t want to use SSH agent.
* **ansible\_ssh\_common\_args** This setting is always appended to the default command line for **sftp**, **scp**, and **ssh**.
* **ansible\_sftp\_extra\_args**This setting is always appended to the default **sftp** command line.
* **ansible\_scp\_extra\_args**This setting is always appended to the default **scp** command line.
* **ansible\_ssh\_extra\_args**This setting is always appended to the default **ssh** command line.
* **ansible\_ssh\_pipelining** Determines whether or not to use SSH pipelining.
* **ansible\_ssh\_executable** This setting overrides the default behavior to use the system **ssh**.

**Privilege escalation**

* **ansible\_become** , allows to force privilege escalation
* **ansible\_become\_method** Allows to set privilege escalation method
* **ansible\_become\_user** allows to set the user you become through privilege escalation
* **ansible\_become\_pass**, allows you to set the privilege escalation password
* **ansible\_become\_exe**, allows you to set the executable for the escalation method selected
* **ansible\_become\_flags**  allows you to set the flags passed to the selected escalation method.

**Remote host environment parameters:**

* **ansible\_shell\_type** the shell type of the target system.
* **ansible\_python\_interpreter** The target host python path. This is useful for systems with more than one Python or not located at **/usr/bin/python**
* **ansible\_\*\_interpreter** Works for anything such as ruby or perl and works just like [ansible\_python\_interpreter](https://docs.ansible.com/ansible/latest/user_guide/intro_inventory.html" \l "ansible-python-interpreter). This replaces shebang of modules which will run on that host.
* **ansible\_shell\_executable** This sets the shell the ansible controller will use on the target machine, overrides executable in ansible.cfg which defaults to **/bin/sh**. You should really only change it if is not possible to use **/bin/sh**

**Non ssh type connection :**

Ansible executes playbooks over SSH but it is not limited to this connection type.

ansible\_connection**:** docker/local

1. **local** This connector can be used to deploy the playbook to the control machine itself.
2. **Docker** This connector deploys the playbook directly into Docker containers using the local Docker client. The following parameters are processed by this connector:

* **ansible\_host** The name of the Docker container to connect to.
* **ansible\_user** The user name to operate within the container. The user must exist inside the container.
* **ansible\_become** If set to true the become\_user will be used to operate within the container.
* **ansible\_docker\_extra\_args** Could be a string with any additional arguments understood by Docker, used to configure a remote Docker daemon to use.

Variables :

used to loop through a set of given values,  Variables are provided through the inventory, by variable files, overwritten on the command line and set in Tower. Variable names should be letters, numbers, and underscores. Variables should always start with a letter.

 extended by including an actual yaml file containing variables: vars.yml , set them true false

in test.yml

---

hosts: all

include\_vars: vars.yml

**vars**:

**favcolor**: blue

**vars\_files**:

- /vars/external\_vars.yml

tasks:

- **name**: using vars

- include\_tasks: dependencies.yml ---- task is yml , when is var from var.yml

when: common\_dep

**ignore\_errors**: True

ansible vault

**ansible-vault create jobagreement.yml – new file**

**ansible-vault encrypt existingfile.yml – existing file**

**ansible-vault view jobagreement.yml -- view encrypted file**

**ansible-vault edit users.yml ---- edit encrypted file**

**ansible-vault rekey jobagreement.yml** -- chnge file pswd

**ansible-vault decrypt jobagreement.yml -- decrypt file**

ansible roles :

Ansible roles you can group your variables, tasks, handlers etc., which increase reusability and most certainly reduce syntax errors. It helps to de-clutter the whole code. In order to create roles, you use the **ansible-galaxy**command which has all the templates to create it.

A role’s directory structure consists of defaults, vars, files, handlers, meta, tasks, and templates

### **defaults**

Within defaults, there is a main.yml file with the default variables used by a role.

1. ---
2. packer\_version: "0.7.1"

### **vars**

variables in vars have a higher priority, which means that they are more difficult to override. Variables in defaults have the lowest priority of any variables available, which means they’re easy to override.