**ANSIBLE**

**Ansible Tutorial**

1) What is Ansible?

2) Configuration Management

3) Push Based v/s Pull Based

4) How to install Ansible

5) Host Inventory

6) Ansible Modules

7) YAML

8) Playbooks

9) Hands On

10) Conclusion

**Configuration Management**

* It is a method through which we automate admin tasks.
* Configuration management tool turns your code into infrastructure.
* So your code would be testable, repeatable and versionable.
* Infrastructure refers to the composite of

Software

Network

People

Process

**ANSIBLE**

* Ansible is one among the DevOps configuration management tools which is famous for its simplicity.
* It is open source software developed by Michael DeHaan and its ownership is on **RedHat**
* Ansible is an open source IT Configuration Management, Deployment & Orchestration tool.
* This tool is very simple to use yet powerful enough to automate complex multi-tier IT application environments.
* Ansible is an automation tool that provides a way to define **Infrastructure as code.**
* **Infrastructure as code (IAC)** simply means that managing infrastructure by writing code rather than using manual processes.
* The best part is that you don’t even need to know the commands used to accomplish a particular task.
* You just need to specify what state you want the system to be in and Ansible will take care of it.
* The main components of Ansible are **playbooks, configuration management and deployment.**
* **Ansible uses playbooks to automate deploy, manage, build, test and configure anything**
* Ansible was written in **Python.**

**Ansible Features**

* Ansible manages machines in an **agent-less manner using SSH**
* Built on top of Python and hence provides a lot of Python's functionality
* **YAML based playbooks**
* Uses SSH for secure connections
* Follows push based architecture for sending configuration related notifications

**Push Based V/s Pull Based**

* Tools like **Puppet and Chef are pull based**
* Agents on the server periodically checks for the configuration information from central server (Master)
* **Ansible is push based**
* Central server pushes the configuration information on target servers.

**What Ansible can do?**

1) Configuration Management

2) App Deployment

3) Continous Delivery

**How Ansible works?**

* Ansible works by connecting to your nodes and pushing out a small program called Ansible modules to them.
* Then Ansible executed these modules and removed them after finished. The library of modules can reside on any machine, and there are no daemons, servers, or databases required.
* The Management Node is the controlling node that controls the entire execution of the playbook.
* The inventory file provides the list of hosts where the Ansible modules need to be run.
* The Management Node makes an SSH connection and executes the small modules on the host’s machine and install the software.
* 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 remove that code in which one was copied on the host machine.
* Ansible basically consists of three components
* Ansible requires the following components in order to automate Network Infrastructure.

1) Controlling Nodes

2) Managed Nodes

3) Ansible Playbook

**Controlling Nodes**

These Network Devices are referred to as the Managed Nodes.

**Managed Nodes**

Managed Nodes are stored in the hosts file for Ansible automation.

**Ansible Playbook**

Ansible **Playbooks** are expressed in **YAML** format and serve as the repository for the various tasks that will be executed on the Managed Nodes (hosts).

Playbooks are a collection of tasks that will be run on one or more hosts.

**Inventory file**

* Ansible's inventory hosts file is used to list and group your servers.
* Its default locaton is /etc/ansible/hosts

**Note: In inventory file we can mention IP address or Hostnames also**

**Few Important Points about Inventory File**

Comments begins with '#' character

Blank lines are ignore

Groups of hosts are delimited by '[header]' elements

You can enter hostnames or ip addresses

**A hostname/ip can be a member of multiple groups**

**Sample Inventory File**

#Blank lines are ignore

#Ungrouped hosts are specifying before any group headers like below

192.168.122.1

192.168.122.2

192.168.122.3

[webservers]

192.168.122.1

#192.168.122.2

192.168.122.3

[dbserver]

192.168.122.1

192.168.122.2

ashokit-db1.com

ashokit-db2.com

**Ansible Setup**

Create 2 Red Hat Systems in AWS (Free Tier Eligible)

* 1 - Ansible System
* 1 - Host System

Connect to all the systems and create ansible user

* $ sudo useradd ansible
* $ sudo passwd ansible
* pwd
* confirm pwd
* $ sudo visudo
* ansible ALL=(ALL) NOPASSWD: ALL
* $ sudo vi /etc/ssh/sshd\_config

comment PasswordAuthentication no

un-comment PasswordAuthentication yes

Restart the server

* $ sudo service sshd restart

**Note: Do the above steps in all the 2 machines**

**Install Ansible in Control Node**

Switch to Ansible user

* $ sudo su ansible

Install Python

* $ sudo yum install python3 -y

Check python version

* $ python --version (it will fail bcz we used python3)
* $ python3 --version

Update python alternatives

* $ sudo alternatives --set python /usr/bin/python3

Check python version

* $ python --version

Install PIP (It is a python package manager)

* $ sudo yum -y install python3-pip

Install Ansible using Python PIP

* $ pip3 install ansible –user

Verify ansible version

* $ ansible --version

Create ansible folder under /etc

* $ sudo mkdir /etc/ansible

Create ansible.cfg file under /etc/ansible and paste complete content from below git link.

Open: **https://raw.githubusercontent.com/ansible/ansible/devel/examples/ansible.cfg**

Copy the content and paste it in ansible.cfg file with below command

* $ sudo vi /etc/ansible/ansible.cfg

Create hosts file under /etc/ansible. Sample content can found in below git link

Open: **https://raw.githubusercontent.com/ansible/ansible/devel/examples/hosts**

Copy the content and paste it in hosts file with below command

* $ sudo vi /etc/ansible/hosts

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* With this Ansible setup completed \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**Generate SSH Key in Control Node and Copy SSH key into Host Node**

1) Now generate SSH key in Ansible Server (Control Node):

* $ sudo su ansible
* $ ssh-keygen

2) Copy it to Host servers as ansible user:

$ ssh-copy-id ansible@<Host-Private-IP>

Ex : $ ssh-copy-id ansible@172.31.43.23

**Note:** Repeat below command by updating HOST IP for all the HOST Servers.

3) Update Host Inventory in Ansible Server to add host servers details.

4) goto the location /etc/ansible/hosts

* $ vi /etc/ansible/hosts

HOST-NODE-IP <172.31.43.23>

5) Use ping module to test Ansible and after successful run you can see the below

output.

* $ ansible all -m ping

**Ansible AD-HOC Commands**

Switch to ansible user and run ansible ad-hoc commands

* $ sudo su ansible

To run any ansible command we will follow below syntax

**# ansible [ all / groupName / HostName / IP ] -m <<Module Name>> -a <<args>>**

Note: Here -m is the module name and -a is the arguments to module.

Example:

It will display date from all host machines.

* **$ ansible all -m shell -a date**

It will display uptime from all host machines.

* **$ ansible all -m shell -a uptime**

There are two default groups, all and ungrouped. All contains every host. Ungrouped contains all hosts that don’t have another group

It will display the all the modules available in Ansible.

* **$ ansible-doc -l**

To display particular module information

* **$ ansible-doc <moduleName>**

To display shell module information

* **$ ansible-doc shell**

To display details of copy module

* **$ ansible-doc -l | grep "copy"**

It will display more information about yum module

* **$ ansible-doc yum**

**Ping Module**

It will ping all the servers which you have mentioned in **inventory file (/etc/ansible/hosts)**

* **$ ansible all -m ping**

It will display the output in single line.

* **$ ansible all -m ping -o**

Redhat release of all the machines

* **$ ansible all -m shell -a 'cat /etc/\*release'**

Kind of mount on all the machines

* **$ ansible all -m shell -a 'mount'**

Check the service status on all the machines

* **$ ansible all -b -m shell -a 'service sshd status'**

Here it will check the disk space use for all the nodes which are from dbservers group

* **$ ansible dbservers -a "df -h"**

Here it will check the disk space use for all the nodes which are from webservers group

* **$ ansible webservers -a "free -m"**

Here it will display date from from webservers group

* **$ ansible webservers -a "date"**

**Yum Module**

It will install vim package in all node machine which you have mentioned in host inventory file.

* **$ ansible all -b -m yum -a "name=vim"**

Check git version in all machines

* **$ ansible all -m shell -a "git --version"**

To install git client in all node machines

* **$ ansible all -m shell -b -a "yum install git -y"**

To installl git only in webserver nodes

* **$ ansible webservers -m shell -b -a "yum install git -y"**

To install webserver only in particular machine

* **$ ansible 172.1921.1.0 -m shell -b -a "yum install git -y"**

$ ansible all -m shell -b -a "name=git state=present"

$ ansible all -m shell -b -a "name=git state=latest"

$ ansible all -m shell -b -a "name=git state=absent"

**present: install**

**latest: update to latest**

**absent: un-install**

To install any software in **ubuntu** server then we should use **apt** package manager

* **$ ansible all -m apt -a "name="git state="present"**

To install httpd package in all node machines

* **$ ansible all -b -m yum -a "name=httpd state=present"**

**Note: Here state=present, is not a mandatory, it is by default.**

To update httpd package in all node machines.

* **$ ansible all -b -m yum -a "name=httpd state=latest"**

To remove httpd package in all node machines.

* **$ ansible all -b -m yum -a "name=httpd state=absent"**
* **$ ansible all -m copy -a "src="index.html dest=/var/www/html/index.html"**

Start httpd service

* **$ ansible all -b -m service -a "name=httpd state=started"**
* **$ ansible all -b -m shell -a "service httpd start"**

**Note: For privilege escalations we can use -b option**

Q) Irrespective of underlying OS which module we can use to manage packages (softwares) using package manager in Ansible?

Ans) Ansible introduced "**package manager**" to work with underlying package manager

**YAML (Yet another Markup Language)**

* YAML Ain’t markup language
* We can make use of this language to store data and configuration in a human-readable format.
* YAML files will have **.yml as an extension**
* Official Website: https://yaml.org/

**Key-Value Pair**

Fruit: Apple

Vegetable: Carrot

Liquid: Water

Meet: Chicken

**Array/List**

Fruits:

- Orange

- Apple

- Banana

- Guava

**Vegetables:**

- Carrot

- Cauliflower

- Tomoto

Here - dash indicate the element of any array.

**PLAYBOOKS**

* Playbook is a **single YAML file**, containing one or more ‘plays’ in a list.
* Plays are ordered sets of tasks to execute against host servers from your inventory file.
* Play defines a set of activities (tasks) to be run on hosts.
* Task is an action to be perform on the host

Examples are

a) Execute a command

b) Run a shell script

c) Install a package

d) Shutdown/Restart the hosts

**Note:** Playbooks start with the YAML three dashes (---) and end with …

Playbook contains the following sections:

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 and can declare all the variables needed in the playbook. We will look at some examples as well.

4) Tasks section – This 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.

**Playbook to Ping All Host Nodes**

---

- hosts: all

tasks:

- name : Test connection

ping:

remote\_user: ansible

hosts: The tasks will be executing in specified group of servers.

name: which is the task name that will appear in your terminal when you run the playbook.

remote\_user: This parameter was formerly called just user. It was renamed in Ansible 1.4 to make it more distinguishable from the user module (used to create users on remote systems).

**Note:** Remote users can also be defined per task.

**Run the playbook Using below command**

* **$ ansible-playbook <<Playbbok file name>>**

It will run the playbook.yml playbook in verbose

* **$ ansible-playbook playbook.yml -v**
* **$ ansible-playbook playbook.yml -vv**
* **$ ansible-playbook playbook.yml -vvv**

It will provide help on ansible playbook command

* **$ ansible-playbook --help**

It will check the syntax of a playbook

* **$ ansible-playbook playbook.yml --syntax-check**

It will do in dry run.

* **$ ansible-playbook playbook.yml --check**

It will display the which hosts would be effected by a playbook before run

* **$ ansible-playbook playbook.yml --list-hosts**

It execute one-step-at-a-time, confirm each task before running with (N)o/(y)es/(c)ontinue

* **$ ansible-playbook playbook.yml --step**

**Install HTTPD + copy index.html + Start Service**

---

- hosts: all

become: true

tasks:

- name: Install Httpd

yum:

name: httpd

state: present

- name: Copy index.html

copy:

src: index.html

dest: /var/www/html/index.html

- name: Start Http Server

service:

name: httpd

state: started

**VARIABLES**

- hosts: all

become: true

tasks:

- name: Install Httpd

yum:

name: "{{package\_name}}"

state: present

- name: Copy index.html

copy:

src: index.html

dest: /var/www/html/index.html

- name: Start Http Server

service:

name: "{{package\_name}}"

state: started

**We can pass variable value in run time like below**

* **$ ansible-playbook filename.yml --extra-vars package\_name=httpd**

**we can define variables with in the playbook**

---

- hosts: all

become: true

vars:

package\_name: httpd

tasks:

- name: Install Httpd

yum:

name: "{{package\_name}}"

state: present

- name: Copy index.html

template:

src: index.html

dest: /var/www/html/index.html

- name: Start Http Server

service:

name: "{{package\_name}}"

state: started

**Group Variables**

For webservers i want to install git-2.1

For dbservers i want to install git-2.3

We can achieve these using group variables

**group vars files should be created at host inventory location**

host-inventory location : **/etc/ansible**

group\_vars/all.yml

group\_vars/<groupName>.yml

Ex: **group\_vars/webservers.yml**

**group\_vars/dbservers.yml**

**group\_vars/all.yml**

/etc/ansible/group\_vars/all.yml

Create mkdir /etc/ansible/group\_vars

vi /etc/ansible/group\_vars/all.yml

test: abc

vi /etc/ansible/group\_vars/appservers.yml

test: xyz

Note: as we have group vars file for appservers it will consider xyz for appservers group and for remaining servers group it will take all.yml file value

**Host variables**

Server specific variables

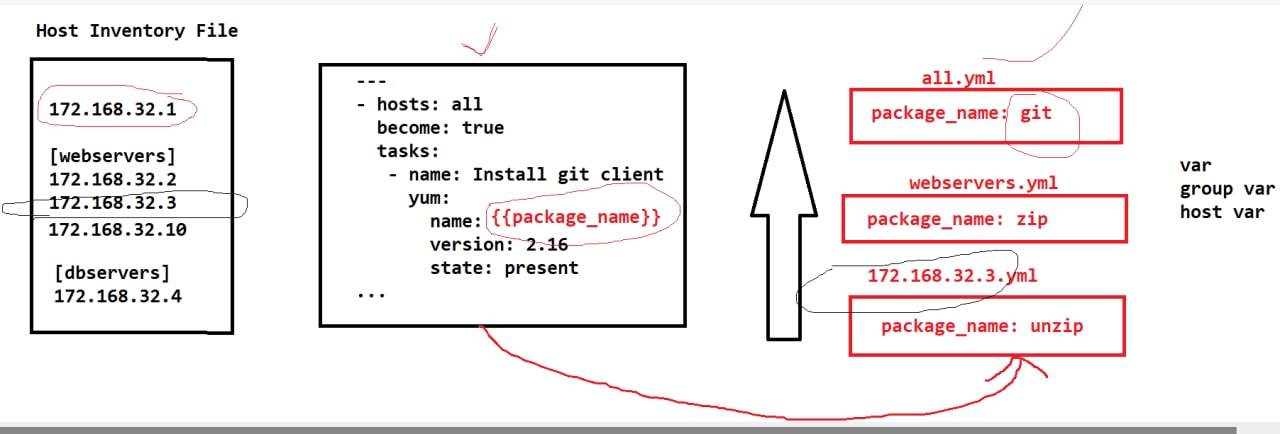
In one group we will have multiple servers

For every host if we want separate variables then we should go for host variables

-> mkdir /etc/ansible/host\_vars

-> create a file with host name or ip

-> vi /etc/ansible/host\_vars/172.138.1.1.yml



**ANSIBLE VAULT**

ansible-vault create <filename>.yml

ansible-vault encrypt <filename>.yml

ansible-vault view <filename>.yml

ansible-vault edit <filename>.yml

ansible-vault decrypt <filename>.yml

ansible-vault rekey <filename>.yml

* To encrypt a playbook we need to set one vault password
* While executing playbook we need to pass vault password
* **$ ansible-playbook <filename>.yml --ask-vault-pass**
* You can store vault password in a file and you can give that file as input to execute playbook
* **$ vi valutpass**
* **$ ansible-playbook filename.yml --vault-password-file=~/vaultpass**

We can see encrypted file in human readable format

* **$ ansible-vault view /etc/ansible/group\_vars/all.yml**

We can edit encrypted file in human readable format

* **$ ansible-vault edit /etc/ansible/group\_vars/all.yml**

We can decrypt the file

* **$ ansible-vault edit /etc/ansible/group\_vars/all.yml**

To update vault password we can use rekey

* **$ ansible-vault rekey /etc/ansible/group\_vars/all.yml**

**HANDLERS AND TAGS**

* Sometimes you want a task to run only when a change is made on a machine. For example, you may want to restart a service if a task updates the configuration of that service, but not if the configuration is unchanged. Ansible uses handlers to address this use case. **Handlers are tasks that only run when notified.**
* **If you have a large playbook, it may become useful to be able to run only a specific part of it rather than running everything in the playbook**. Ansible supports a “tags:” attribute for this reason.
* Tags can be applied tomany structures in Ansible, but its simplest use is with individual tasks.

---

- hosts: all

become: true

gather\_facts: no

vars:

package\_name: httpd

tasks:

- name: install httpd

yum:

name: "{{package\_name}}"

state: present

- name: Copy index.html

copy:

src: index.html

dest: /var/www/html/

notify:

Start Httpd Server

tags:

- copy

- name: debug message

msg: Copy completed

tags:

- debug

- install

handlers:

- name: Start Httpd Server

service:

name: {{package\_name}}

state: started

To display all tags available in playbook

* **$ ansible-playbook filename.yml --list-tags**
* **$ ansible-playbook filename.yml --tags "install"**
* **$ ansible-playbook filename.yml --tags "debug,copy"**
* **$ ansible-playbook filename.yml --skip-tags**

**Installing Multiple Softwares**

- hosts: all

tasks:

- name: install softwares

yum:

name: "{{item}}"

state: present

with\_items:

- wget

- zip

- unzip

**Another approach**

- hosts: all

tasks:

- name: install softwares

yum:

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

state: present

Scope of the variable is within the playbook. We can refer vars in the tasks or templates which we are using the playbook.

- hosts: all

tasks:

- name: vars demo

debug:

msg: "var test value: {{test}}"

- name: copy file

template:

src: test.txt

dest: /tmp/test.txt

- name: cat the file

shell: "cat /tmp/test.txt"

register: output

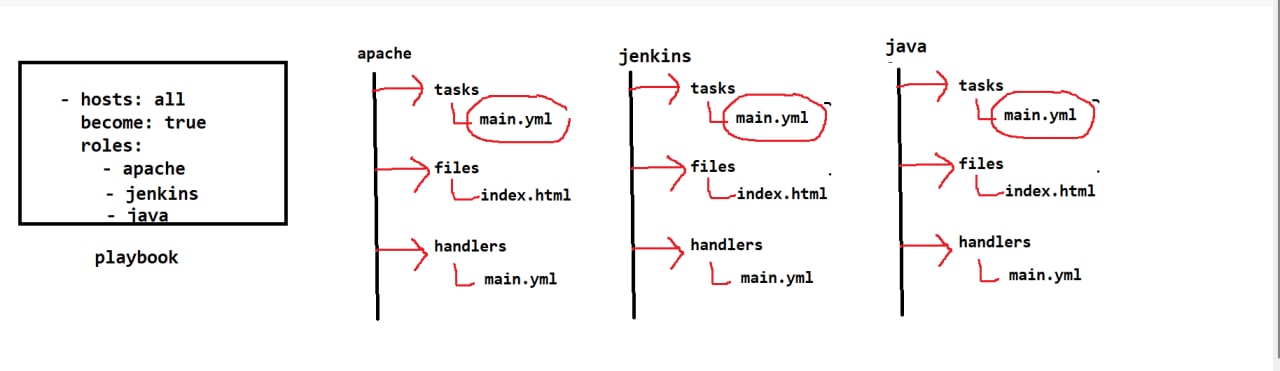
- name: print file content

debug:

msg: "File Content {{output.stdout}}"

**ANSIBLE ROLES**

* Roles are a level of abstraction on top of tasks and playbooks that let you structure your Ansible configuration in a modular and reusable format
* **As you add more and more functionality to your playbooks, they can become difficult to maintain**
* **Roles allow you to break down a complex playbook into separate, smaller chunks** that can be coordinated by a central entry point.



# Sample playbook with Role

---

- hosts: all

become: true

roles:

- apache

1. Ansible roles are consists of many playbooks, which is similar to modules in puppet and cook books in chef. We term the same in ansible as roles.

2. Roles are a way to group multiple tasks together into one container to do the automation in very effective manner with clean directory structures.

3. Roles are set of tasks and additional files for a certain role which allow you to break up the configurations.

4. It can be easily reuse the codes by anyone if the role is suitable to someone.

5. It can be easily modify and will reduce the syntax errors.

**How do we create Ansible Roles?**

* To create an Ansible role, use **"ansible-galaxy"** command which has the templates to create it.
* **$ sudo su ansible**
* **$ cd /home/ansible**
* **$ mkdir roles**
* **$ ansible-galaxy init roles/apache**
* Where, ansible-glaxy is the command to create the roles using the templates.
* init is to initiliaze the role.
* apache is the name of the role
* List out the directory created under roles/apache
* **$ sudo yum install tree**
* **$ tree roles/apache/**

We have got the clean directory structure with the ansible-galaxy command. Each directory must contain a main.yml file, which contains the relevant content.

**Directory Structure:**

tasks – contains the main list of tasks to be executed by the role.

handlers – contains handlers, which may be used by this role or even anywhere outside this role.

defaults – default variables for the role.

vars – other variables for the role. Vars has the higher priority than defaults.

files – contains files required to transfer or deployed to the target machines via this role.

templates – contains templates which can be deployed via this role.

meta – defines some data / information about this role (author, dependency, versions, examples, etc,.)

**Lets take an example to create a role for Apache Web server.**

Below is a sample playbook codes to deploy Apache web server. Lets convert this playbook codes into Ansible role.

- hosts: all

become: true

tasks:

- name: Install Httpd

yum:

name: httpd

state: present

- name: Copy index.html

template:

src: index.html

dest: /var/www/html/index.html

- name: Start Http Server

service:

name: httpd

state: started

First, move on to the Ansible roles directory and start editing the yml files.

**$ cd roles/apache**

**1. Tasks**

Edit main.yml available in the tasks folder to define the tasks to be executed.

**$ vi tasks/main.yml**

---

# tasks file for roles/apache

- name: install httpd

yum:

name: httpd

state: present

- name: Copy index.html

copy:

src=index.html

dest=/var/www/html/

notify:

- restart apache

- name: restart apache

service:

name=httpd

state=restarted

**2. Files**

Copy required files into files directory or create index.html file with content

**3. Handlers**

Edit handlers main.yml to restart the server when there is a change. Because we have already defined it in the tasks with notify option. Use the same name “restart apache” within the main.yml file as below.

**$ cat handlers/main.yml**

- name: restart apache

service:

name: httpd

state: restarted

We have got all the required files for Apache roles. Lets apply this role into the ansible playbook “runsetup.yml” as below to deploy it on the client nodes.

**$ cat /home/ansible/runsetup.yml**

---

- hosts: all

become: true

roles:

- apache

...

Execute playbook which contains apache role

**$ ansible-playbook runsetup.yml**

If you have created **multiple roles**, you can use the below format to add them in the playbook

---

- hosts: all

become: true

roles:

**- apache**

**- jenkins**

**- java**

**- maven**

**- sonar**

What We Covered in Ansible?

1) What is Configuration Management?

2) What is Ansible?

3) Advantages of Ansible

4) Push Based V/s Pull Based Mechanism

5) Ansible Installation

6) Ansible Architecture

7) Host Inventory File

8) Host Groups in Inventory

9) Ansible Ad Hoc Commands

10) YAML

11) Working with YAML in VS Code IDE

12) Playbook Introduction

13) Playbook Commands

14) Writing Playbooks

15) Variables

- Runtime variables

- Playbook variables

- Group variables

- Host variables

16) Handlers

17) Tags

18) Roles

19) Ansible Vault

20) Playbook for Multiple OS Family Based Hosts

**ANSIBLE TOWER**

* Ansible Tower is a **web based GUI** tool which is used to manage infrastructure configurations.
* We can centralize our infrastructure configurations using this UI
* We can do job scheduling
* It is providing graphical inventory management

**Ansible Tower features**

1) Clean dashboard

2) Manage Iventory Dynamically

3) Realtime Job Status

4) Job Scheduling

5) Integrate Internal Notification

6) Rolebase access control (RABAC)

7) Audit Job and Tower Resource

8) Store Credentials safely

9) REST API Integration

10) Self Service UI

**Pre-Requisites to Install Ansible Tower**

* Red Hat Enterprise Linux 6 64-bit
* 20 GB Hard Disk
* 4 GB Ram
* Note: m4.large instance required

**Step-1: Install Ansible Software**

1. Update package manager

* **$ sudo yum -y update**

2. Install epel package manager

* **$ sudo dnf install https://dl.fedoraproject.org/pub/epel/epel-release-latest-8.noarch.rpm -y**

3. Check epl repositories install

* **$ sudo dnf repolist epel**

4. Ansible tower uses ansible playbook to deploy so we should install Ansible first

* **$ sudo yum -y install ansible vim curl**

**Step-2: Download Ansible Tower Archieve Folder**

* **$ mkdir /tmp/tower && cd /tmp/tower**
* **$ curl -k -O https://releases.ansible.com/ansible-tower/setup/ansible-tower-setup-latest.tar.gz**
* **$ ls**

Extract tar file

* **$ tar xvf <tar-file>**

Go to extracted folder and update admin\_pwd and postgres\_db pwd in inventory file

* **$ vim inventory**

Run the script file to run playbook

* **$ sudo ./setup.sh**

**Note: It will take 2 to 3 minutes**

check ansible tower status

* **$ sudo ansible-tower-service status**

Start Ansible Tower

* **$ sudo ansible-tower-service start**

**Note: Open 443 port because Ansible Tower Running on HTTPS**

Access EC2 VM IP in browser and login

default uname : admin

pwd : we have given in inventory file (ashokit)

**Note: This is enterprise product so we need license. We can get trail version**

Create account in red hat and request for license

www.redhat.com/en/technologies/management/ansible/try-it

Click on 'Request license' -> Complete process

(https://access.redhat.com/management/subscriptions)

Loing with redhat account credentials

**Configure 2 Host Nodes IN ec2**

1) Login into tower

2) Create Groups and add hosts

3) Create credentials with Machine and provide ssh private key

Create Project

Give SCM as git repo url of playbooks