**Introduction**

Ansible is an open source IT configuration management, deployment, and orchestration tool. It is unique from other management tools in many respects, aiming to provide large productivity gains to a wide variety of automation challenges as a more productive drop-in replacement for many core capabilities in other automation solutions.

Ansible is designed to be minimal in nature, consistent, secure, and highly reliable, with an extremely low learning curve for administrators, developers, and IT managers. Ansible seeks to keep descriptions of IT easy to build, and easy to understand - such that new users can be quickly brought into new IT projects.

**ARCHITECTURE, AGENTS, AND SECURITY**

One of the primary differentiators between Ansible and many other tools in this space is its architecture. Ansible is an agentless tool that runs in a ‘push’ model - no software is required to be installed on remote machines to make them manageable. Ansible by default manages remote machines over SSH (Linux and UNIX) or WinRM (Windows), using the remote management frameworks that already exist natively on those platforms.

Ansible builds on this by not requiring dedicated users or credentials - it respects the credentials that the user supplies when running Ansible. Similarly, Ansible does not require administrator access, leveraging sudo, su, and other privilege escalation methods on request when necessary.

This method allows Ansible to be more secure. By using the credentials passed by the user, those with access to the control server (or source control) cannot make content be pushed out to remote systems (or otherwise command them) without also having credentials on remote systems.

Similarly, by operating in a push-based model where only needed code (called Ansible ‘modules’) are passed to remote machines, remote machines cannot see or affect how other machines are configured.

By running in an agentless manner, no resources are consumed on managed machines when Ansible is not managing them. These attributes together make Ansible ideal for high-security environments or high-performance cases where there are concerns about stability or permanence of a management agent, but are generally useful attributes in all computing areas.



**Exercise 1: Ansible Installation**

- Update the apt package index:

$ sudo apt-get update

- Install the *latest version* of Ansible CE

$ sudo apt-get install ansible -y

- Let’s check the Ansible installation.

root@master:~# ansible --version

ansible 2.0.0.2

config file = /etc/ansible/ansible.cfg

configured module search path = Default w/o overrides

root@master:~#

- Install Ansible SSH Dependency

root@master:~# apt-get install sshpass -y

- Enable SSH Access on Agent’s / Worker Node’s

root@worker01:~# egrep 'PermitRootLogin|PasswordAuthentication' /etc/ssh/sshd\_config | grep -v "^#"

PermitRootLogin yes

PasswordAuthentication yes

root@worker01:~#

root@worker01:~# service sshd restart

- Reset the root user password.

root@worker01:~# passwd root

Enter new UNIX password:

Retype new UNIX password:

passwd: password updated successfully

root@worker01:~#

- Verify SSH Access with all the respective node’s

root@master:~# ssh root@172.31.0.11 ls -ltr

root@172.31.0.11's password:

-rw-r--r-- 1 root root 0 Apr 26 10:16 nohup.out

root@master:~#

**Exercise 2: Inventory**

The Ansible inventory file defines the hosts and groups of hosts upon which commands, modules, and tasks in a playbook operate. The file can be in one of many formats depending on your Ansible environment and plugins. The default location for the inventory file is **/etc/ansible/hosts**. If necessary, you can also create project-specific inventory files in alternate locations.

- Let’s create basic inventory:

root@master:~/myproject/ansible/demo1# cat inventory

172.31.0.11

172.31.0.12

root@master:~/myproject/ansible/demo1#

- Group, Group of Groups & Variables

root@master:~/myproject/ansible/demo2# cat inventory

[web]

172.31.0.11

172.31.0.14

[db]

172.31.0.12

172.31.0.13

[prod]

172.31.0.11

172.31.0.12

[qa]

172.31.0.13

172.31.0.14

[dc:children]

prod

qa

[dc:vars]

ansible\_python\_interpreter=/usr/bin/python2.7

ansible\_ssh\_user=root

ansible\_ssh\_pass=redhat

root@master:~/myproject/ansible/demo2#

- Run ansible ad-hoc command in order to test ansible communication:

root@master:~/myproject/ansible/demo2# ansible all -i inventory -m ping

172.31.0.11 | SUCCESS => {

"changed": false,

"ping": "pong"

}

- Run ansible ad-hoc command with **PROD** Group with **Shell** Module:

root@master:~/myproject/ansible/demo2# ansible prod -i inventory -m shell -a 'ls -ltr'

172.31.0.11 | SUCCESS | rc=0 >>

-rw-r--r-- 1 root root 0 Apr 26 10:16 nohup.out

172.31.0.12 | SUCCESS | rc=0 >>

-rw-r--r-- 1 root root 0 Apr 26 10:16 nohup.out

root@master:~/myproject/ansible/demo2#

- Group Combination Operation – **Union**

root@master:~/myproject/ansible/demo2# ansible web:prod -i inventory -m shell -a 'ls -ltr'

172.31.0.11 | SUCCESS | rc=0 >>

-rw-r--r-- 1 root root 0 Apr 26 10:16 nohup.out

172.31.0.12 | SUCCESS | rc=0 >>

-rw-r--r-- 1 root root 0 Apr 26 10:16 nohup.out

172.31.0.16 | UNREACHABLE! => {

"changed": false,

"msg": "ERROR! SSH encountered an unknown error during the connection. We recommend you re-run the command using -vvvv, which will enable SSH debugging output to help diagnose the issue",

"unreachable": true

}

root@master:~/myproject/ansible/demo2#

- Group Combination Operation – **Intersection**.

root@master:~/myproject# ansible "web:&prod" -i inventory -m shell -a 'ls -ltr'

172.31.0.11 | SUCCESS | rc=0 >>

-rw-r--r-- 1 root root 0 Apr 26 10:16 nohup.out

root@master:~/myproject#

- Group Combination Operation – **Not**.

root@master:~/myproject# ansible 'web:&prod:!python3' -i inventory -m shell -a 'ls -ltr'

172.31.0.11 | SUCCESS | rc=0 >>

-rw-r--r-- 1 root root 0 Apr 26 10:16 nohup.out

root@master:~/myproject#

**Exercise 3: PlayBooks**

Ansible performs automation and orchestration of IT environments via Playbooks. Playbooks are a YAML definition of automation tasks that describe how a particular piece of automation should be done. Like their namesake, Ansible Playbooks are prescriptive, yet responsive descriptions of how to perform an operation - in this case, IT automation, that clearly states what each individual component of your IT infrastructure needs to do, but still allows components to react to discovered information, and to operate in concert with each other.

Ansible Playbooks consist of series of ‘plays’ that define automation across a set of hosts, known as the ‘inventory’. Each ‘play’ consists of multiple ‘tasks,’ that can target one, many, or all of the hosts in the inventory.

Each task is a call to an **Ansible module - a small piece of code for doing a specific task.** These tasks can be simple, such as placing a configuration file on a target machine, or installing a software package. They can be complex, such as spinning up an entire CloudFormation infrastructure in Amazon EC2.

Ansible includes hundreds of modules, ranging from simple configuration management, to managing network devices, to modules for maintaining infrastructure on every major cloud provider.

- Let’s create out first PlayBook.

root@master:~/myproject/ansible/demo3# cat webserver.yaml

---

- hosts: web

sudo: yes

tasks:

- name: Install Apache Packages

apt: name=apache2 state=present

- name: Start Apache Service

service: name=apache2 state=started

root@master:~/myproject/ansible/demo3#

- Run newly created Playbook:

root@master:~/myproject/ansible/demo3# ansible-playbook -i inventory webserver.yaml

PLAY \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

TASK [setup] \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

ok: [172.31.0.11]

TASK [Install Apache Packages] \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

ok: [172.31.0.11]

TASK [Start Apache Service] \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

ok: [172.31.0.11]

PLAY RECAP \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

172.31.0.11 : ok=3 changed=0 unreachable=0 failed=0

root@master:~/myproject/ansible/demo3#

**Exercise 4: Playbook, Variables & Templates**

Templates are simple text files that we use in Ansible which contains all your configuration parameters. During the playbook execution, depending on the conditions like which cluster you are using, the variables will be replaced with the relevant values. Most of the time we use them to replace configuration files or place some documents on the server. We can do much more than replacing the variables with the help of Jinj2 templating engine. We can have conditional statements, loops, filters for transforming the data, do arithmetic calculations, etc. The template files will usually have the .j2 extension, which denotes the Jinja2 templating engine used.

- Let’s create a templates directory to place our template’s:

root@master:~/myproject/ansible/demo4# mkdir templates

- Let’s create static & j2 template file which consist of **user define & dynamic** Variables:

root@master:~/myproject/ansible/demo4/templates# cat index.html

Welcome to the World of Ansible Automation.!!!

root@master:~/myproject/ansible/demo4/templates#

root@master:~/myproject/ansible/demo4/templates# cat info.j2

<!DOCTYPE html>

<html lang="en">

<head>

<meta cherset="utf-8">

</script>

</head>

<body background="mario.jpg">

<h1>{{ansible\_hostname}} is managed by Ansible</h1><br>

<h2>Welcome to host : {{ansible\_hostname}}</h2>

<h2>Host IP Address : {{ansible\_all\_ipv4\_addresses}}</h2>

<h2>OS Family : {{ansible\_os\_family}}</h2>

<h2>Running Since : {{ansible\_uptime\_seconds}}</h2>

<h2>Document Root : {{doc\_root}}</h2>

<h2>Brought to you by : {{name}}</h2>

</body>

</html>

root@master:~/myproject/ansible/demo4/templates#

- Update the playbook with Variable definition & Template Deployments:

root@master:~/myproject/ansible/demo4# cat webserver.yaml | grep -v "^$"

---

- hosts: web

sudo: yes

vars:

http\_port: 80

doc\_root: "/var/www/html"

name: Jenkins-CI\_CD

tasks:

- name: Install Apache Packages on Debian

apt: name=apache2 state=present

when: ansible\_os\_family == "Debian"

- name: Start Apache Service on Debian

service: name=apache2 state=started

when: ansible\_os\_family == "Debian"

- name: Install Apache Packages on CentOS

yum: name=httpd state=present

when: ansible\_os\_family == "RedHat"

- name: Start Apache Service on CentOS

service: name=httpd state=started

when: ansible\_os\_family == "RedHat"

- debug: msg="This is {{name}} what you supplied"

- name: Deploy Custom Apache Config.

template: src=templates/apache2.j2 dest=/etc/apache2/apache2.conf

notify:

- Restart Apache

- name: Create Site Location

file: path={{doc\_root}} state=directory

- name: Download Super Mario Background Image.

get\_url: url="http://www.mariomayhem.com/downloads/wallpapers/2/nsmb\_wp2\_1024.jpg" dest={{doc\_root}}/mario.jpg mode=0755

- name: Deploy Index File

copy: src=templates/index.html dest={{doc\_root}}/index.html

- name: Deploy Info. File

template: src=templates/info.j2 dest={{doc\_root}}/info.html

handlers:

- name: Restart Apache

service: name=apache2 state=restarted

root@master:~/myproject/ansible/demo4#

- Run the Playbook:

root@master:~/myproject/ansible/demo3# ansible-playbook -i inventory webserver.yaml

**Exercise 5: Roles**

Ansible also supports encapsulating Playbook tasks into reusable units called ‘roles.’ Ansible roles can be used to easily apply common configurations in different scenarios, such as having a common web server configuration role that may be used in the development, test, and production automation. The Ansible Galaxy community site contains thousands of roles that can be used and customized to build Playbooks.

- Let’s create a roles directory to place our template’s:

root@master:~/myproject/ansible/demo5# mkdir roles

- Create a new webservers role inside roles directory:

root@master:~/myproject/ansible/demo6# ansible-galaxy init webservers -p roles/

- webservers was created successfully

root@master:~/myproject/ansible/demo6#

- Create a new webservers role inside roles directory:

root@master:~/myproject/ansible/demo6# ls -l roles/webservers

drwxr-xr-x 2 root root 4096 May 3 18:40 defaults

drwxr-xr-x 2 root root 4096 May 3 18:40 files

drwxr-xr-x 2 root root 4096 May 3 18:40 handlers

drwxr-xr-x 2 root root 4096 May 3 18:41 meta

-rw-r--r-- 1 root root 1328 May 3 18:40 README.md

drwxr-xr-x 2 root root 4096 May 3 18:41 tasks

drwxr-xr-x 2 root root 4096 May 3 18:41 templates

drwxr-xr-x 2 root root 4096 May 3 18:41 tests

drwxr-xr-x 2 root root 4096 May 3 18:41 vars

root@master:~/myproject/ansible/demo6#

- Generate Webserver Role with ansible-galaxy command:

root@master:~/myproject/ansible/demo6# ansible-galaxy init webservers -p roles/

- webservers was created successfully

root@master:~/myproject/ansible/demo6#

- Copy all the Task from the previous playbook into webserver roles tasks/main.yaml:

root@master:~/myproject/ansible/demo6# cat roles/webservers/tasks/main.yml | grep -v "^$"

---

# Tasks file for webservers

- name: Install Apache Packages on RedHat Nodes

yum: name=httpd state=present

when: ansible\_os\_family == "RedHat"

- name: Install Apache Packages on Ubuntu Nodes

apt: name=apache2 state=present

when: ansible\_os\_family == "debain"

- name: Start Apache Service

service: name=httpd state=started

- name: Checking the Document Root Avalabilty

command: ls {{doc\_root}}

register: result1

ignore\_errors: yes

- debug: msg="Unable to find Document Root"

when: result1|failed

- name: Create Document Root Dir. Location

file: path={{doc\_root}} state=directory

when: result1|failed

- name: Deploy Custom Apache Configuration

template: src=templates/httpd.j2 dest=/etc/httpd/conf/httpd.conf

notify:

- Restart Apache

- amit

- name: Deploy Sample Sites Files

template: src=templates/index.j2 dest={{doc\_root}}/index.html

notify:

- nothing

- name: Deploy Sample Sites Files

template: src=templates/info.j2 dest={{doc\_root}}/info.html

- debug: msg="Username is varible prompt is{{username}}"

root@master:~/myproject/ansible/demo6#

- Copy Handlers from the previous playbook into webserver roles handlers/main.yaml:

root@master:~/myproject/ansible/demo6# cat roles/webservers/handlers/main.yml | grep -v "^$"

---

# handlers file for webservers

- name: Restart Apache

service: name=httpd state=restarted

- name: nothing

service: name=httpd state=restarted

- name: amit

command: 'echo "Hello Handlers" > {{doc\_root}}/abc.html'

root@master:~/myproject/ansible/demo6#

- Copy Variables from previous playbook into webserver roles vars/main.yaml:

root@master:~/myproject/ansible/demo6# cat roles/webservers/vars/main.yml | grep -v "^$"

---

# vars file for webservers

http\_port: 80

doc\_root: /var/www/html/vashist

doc\_dir: /vashist/

username: "Amit Vashist"

root@master:~/myproject/ansible/demo6#

- Copy templates from previous playbook into webserver roles templates directory:

root@master:~/myproject/ansible/demo6# ls -ltr roles/webservers/templates/

-rw-r--r-- 1 root root 486 May 3 18:48 info.j2

-rw-r--r-- 1 root root 47 May 3 18:48 index.html

-rw-r--r-- 1 root root 7188 May 3 18:48 apache2.j2

root@master:~/myproject/ansible/demo6#

- Create a playbook which call webserver roles:

root@master:~/myproject/ansible/demo6# cat webserver.yaml

---

- hosts: prod

sudo: yes

roles:

- webservers

root@master:~/myproject/ansible/demo6#

- Create a site file which will be a primary file & responsible respective roles/code deployment:

root@master:~/myproject/ansible/demo6# cat site.yaml

---

- hosts: all

sudo: yes

- include: webserver.yaml

root@master:~/myproject/ansible/demo6#

- Let’s run the ansible site.yaml

root@master:~/myproject/ansible/demo6# ansible-playbook -i inventory site.yaml