++++++++++++++++++++
Ansible Tutorial
++++++++++++++++++++

1) What is Ansible
2) Configuration Management
3) Push Based vs 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 â€”

Software
Network
People
Process

++++++++++++
Ansible
++++++++++++

-> Ansible is one among the DevOps configuration management tools which is famous for its simplicity.

-> It is an 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 donâ€™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 Vs 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 hostâ€™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
++++++++++++++++++
are usually Linux Bastion Servers that are used to access the switches/routers and other Network

Devices.
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
-> Ungrouped hosts are specifying before any group headers like below

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

Sample Inventory File
++++++++++++++++++++
#Blank lines are ignore

#Ungrouped hosts are specifiying 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 3 Red Hat Systems in AWS (Free Tier Eligible - t2.micro)

1 - Control Node
2 - Managed Nodes

=> Connect to all the 3 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 3 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 **********

1) Update Host Inventory in Ansible Server to add host servers details to test connection

$ sudo vi /etc/ansible/hosts

$ Connect using username and password
HOST-NODE-IP ansible_user=ansible ansible_password=password

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

$ ansible all -m ping

Note : Install sshpass in Ansible server if you get below error .

"to use the 'ssh' connection type with passwords, you must install the sshpass
program

$ sudo yum install sshpass

$ ansible all -m ping

++
+++++
Generate SSH Key In Control Node and Copy SSH key into Host Nodes (Managed Nodes)
++
+++++

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

$ sudo su ansible

# Generate ssh key using below command
$ ssh-keygen

2) Copy it to Manaded Nodes as ansible user

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

Ex : $ ssh-copy-id ansible@172.31.33.36

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

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

# vi /etc/ansible/hosts

# Connect using username and password
#192.168.1.105 ansible_user=ansible ansible_password=password

# If ssh keys are copied
172.31.43.23

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

# ping all managed nodes listed in host inventory file
$ ansible all -m ping

#ping only webservers listed in host inventory file
$ ansible webservers -m ping

#ping only dbservers listed in host inventory file
$ ansible dbservers -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 donâ€™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

# it will 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

# Date of all machines
$ ansible all -m shell -a 'date'

# 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 menyioned 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 privilige 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 Ainâ€™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.


name: Ashok
age: 29
phno: 123456
email: ashokitschool@gmail.com
```

```
hobbies:
  - cricket
  - dance
  - singing

# person data in yml

person:
  id: 101
  name: Raju
  email: raju@gmail.com
  address:
    city: Hyd
    state: TG
    country: India
  job:
    companyName: IBM
    role: Tech Lead
    pkg: 25 LPA
  hobbies:
    - cricket
    - chess
    - singing
    - dance



# using --- hypens to seperate the data

---
person:
  id: 101
  name: Raju
  email: raju@gmail.com
  address:
    city: Hyd
    state: TG
    country: India
  job:
    companyName: IBM
    role: Tech Lead
    pkg: 25 LPA
  hobbies:
    - cricket
    - chess
    - singing
    - dance
---
movie:
  name: Bahubali
  hero: Prabhas
  heroine: Anushka
  villian: Rana
  director: SS Rajamouli
  budget: 100cr
...


++++++++++++++++++++
Playbooks
++++++++++++++++++++

-> Playbook is a single YAML file, containing one or more â€˜playsâ€™ 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 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 YML / YAML starts with the three hyphens (---) and ends with three dots (â€¦)

Playbook contains the following sections:

1) Every playbook starts with 3 hyphens â€˜---â€˜

2) Host section â€“ Defines the target machines on which the playbook should run. This is based on the Ansible host inventory file.

3) Variable section â€“ This is optional and can declare all the variables needed in the playbook. We will look at some examples as well.

4) Tasks section â€“ 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
  gather_facts: no
  remote_user: anisble
  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 Httpd 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 also

---

- 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

...



---
- hosts: all
  become: true
  tasks:
   - name: install software
     yum:
       name: "{{package_name}}"
       state: present
...


$ ansible-playbook filename --extra-vars package_name=mysql


=================
Group Variables
=================

-> For webservers i want to install git

-> For dbservers i want to install mysql

-> We can achieve this 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:

$ mkdir /etc/ansible/group_vars

$ sudo vi /etc/ansible/group_vars/webservers.yml
package: git

$ sudo vi /etc/ansible/group_vars/dbservers.yml
package: mysql



============
```

Host vars
==========
-> server specific variables

-> In one group we will have multiple servers

-> For every host if we wan seperate variables then we should go for host vars

-> mkdir /etc/ansible/host_vars

-> create a file with host name or ip

-> vi /etc/ansible/host_vars/172.138.1.1.yml

==========
Variable Value we can declare with in playbook

Variable value we can supply in runtime

Variable value we can declare in hosts_vars

Variable value we can declare in group_vars
==========

==========
Ansible Vault
==========

-> It is used to secure our data

-> When we configure uname and pwd in variables files everybody can see them which is not a good practise

-> When we are dealing with sensitive data then we should secure that data

-> Using Ansible Vault we can protect or secure our data

-> Using Ansible vault we can encrypt and we can decrypt data

Encryption ---> Converting from readable format to un-readbale format

De-Cryption ---> Converting un-readable format to readable format (bringing file back to normal state)

==========
Ansible Vault Commands
==========

$ ansible-vault encrypt <filename>.yml : To encrypt our yml file

$ ansible-vault view <filename>.yml : To see original data from encrypted file

$ ansible-vault edit <filename>.yml : To edit data in original format

$ ansible-vault decrypt <filename>.yml : To decrypt the file (bring the file to normal state)

$ ansible-vault rekey <filename>.yml : To change vault password

```
-> 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 decrypt /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
================

-> Handlers are used to notify the tasks to execute

-> Using Handlers we can execute tasks based on other tasks status

-> To inform the handler to execute we will use 'notify' keyword

-> Using Tag we can map task to a tag-name

-> Using tag name we can execute particular task and we can skip particular task also
```

```
---
- hosts: all
  become: true
  gather_facts: no
  vars:
    package_name: httpd
  tasks:
    - name: install httpd
      yum:
        name: "{{package_name}}"
        state: present
      tags:
        - install
    - name: Copy index.html
      copy:
        src: index.html
        dest: /var/www/html/
       tags:
          - copy
```

```
      notify:
        Start Httpd Server
  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 "install,copy"

$ ansible-playbook filename.yml --skip-tags "install"




++++++++++++++++++++++++++++++
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
```