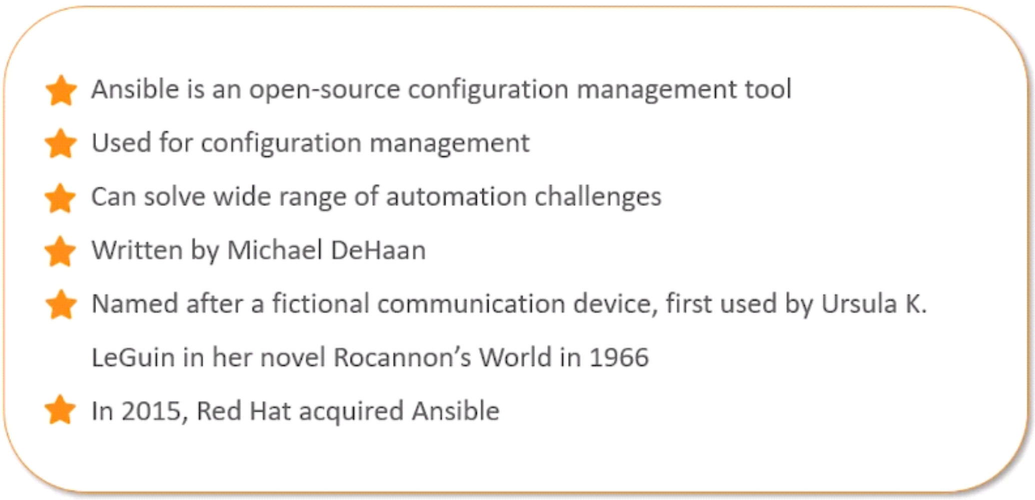
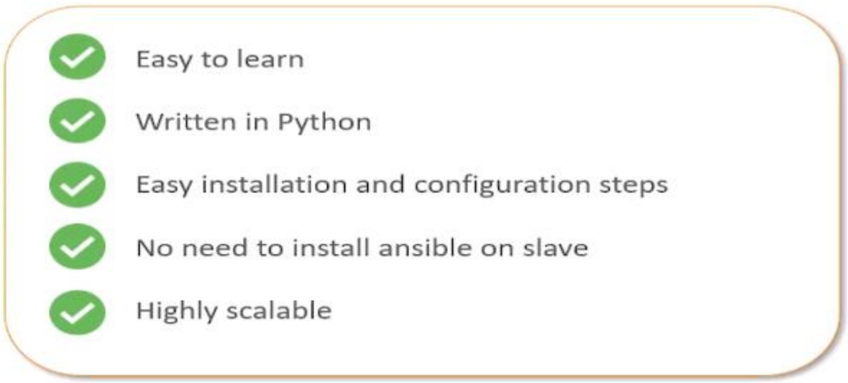
Ansible

Thursday, January 9, 2020 12:47 PM



- Ansible is an open-source configuration management tool
- Used for configuration management
- Can solve wide range of automation challenges
- Written by Michael DeHaan
- Named after a fictional communication device, first used by Ursula K. LeGuin in her novel Rocannon's World in 1966
- In 2015, Red Hat acquired Ansible



- ✓ Easy to learn
- ✓ Written in Python
- ✓ Easy installation and configuration steps
- ✓ No need to install ansible on slave
- ✓ Highly scalable

Companies using Ansible

Apple

NASA

Intel

Percussion

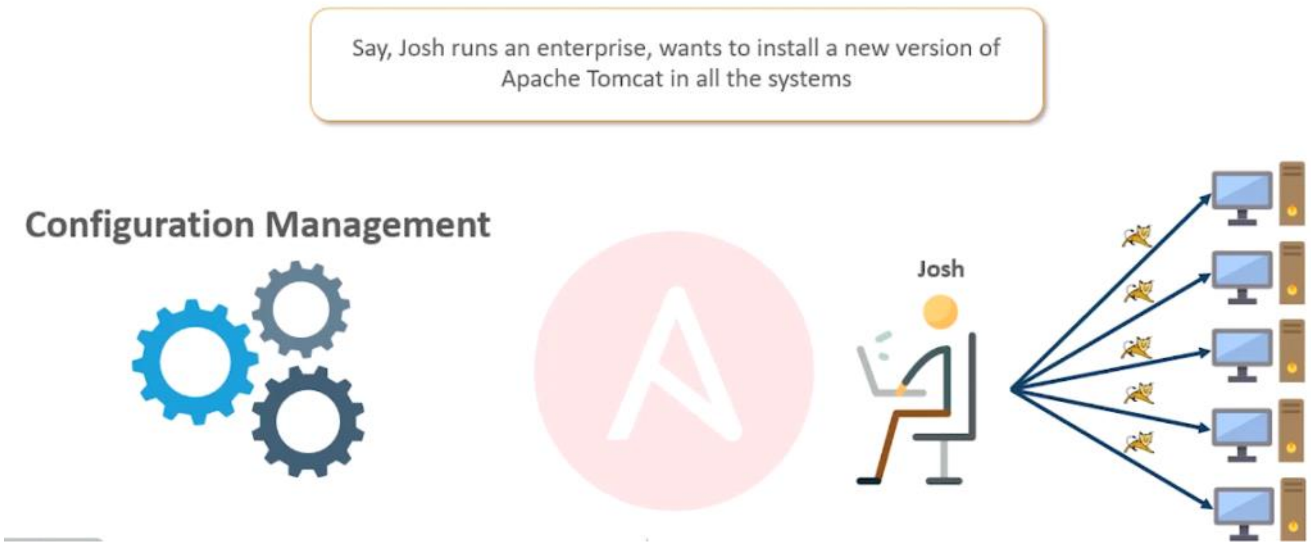
Cisco

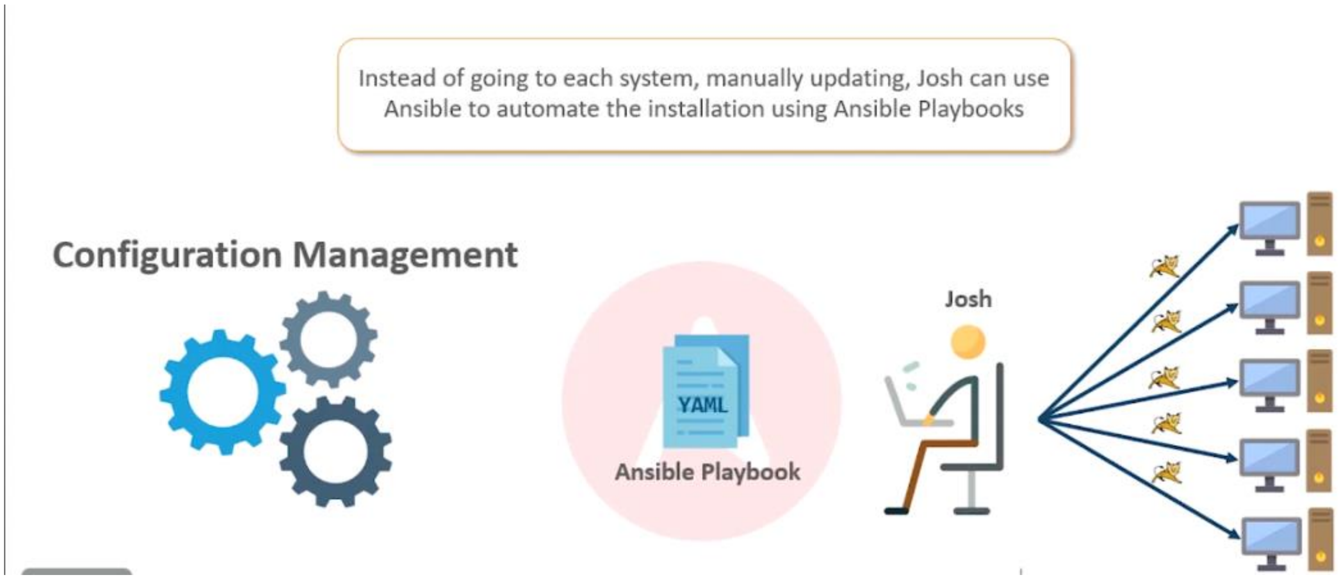
Twitter

How Ansible works

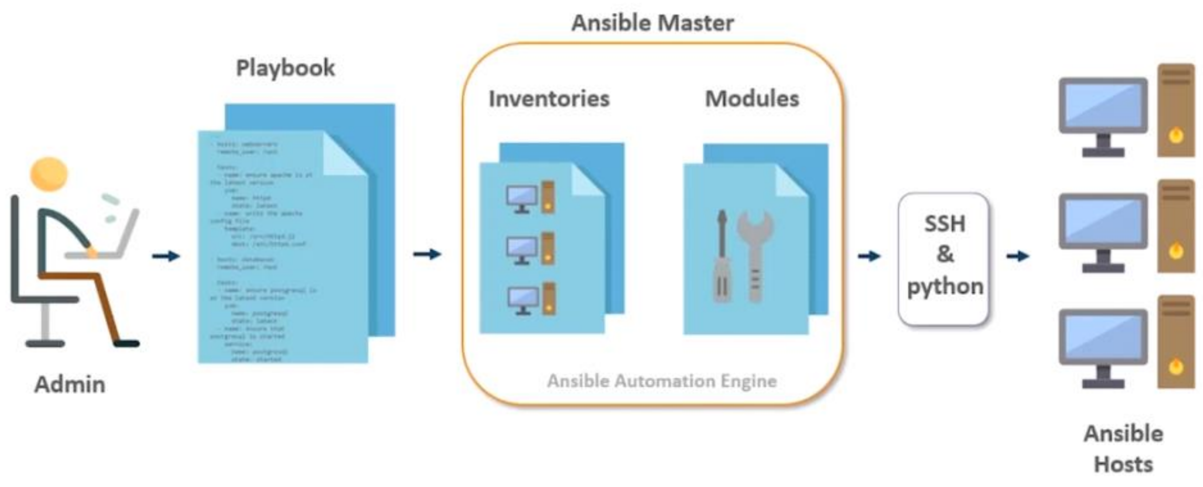
With the help of **Ansible Playbooks**,
which are written in a very simple language, **YAML**

Problem Statement

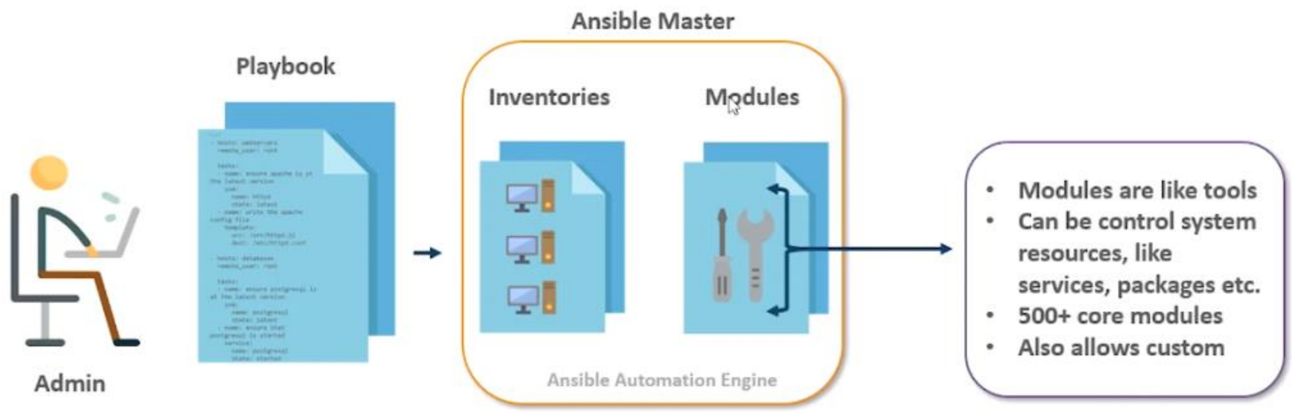

Say, Josh runs an enterprise, wants to install a new version of Apache Tomcat in all the systems


Instead of going to each system, manually updating, Josh can use Ansible to automate the installation using Ansible Playbooks
Configuration Management
YAML
Ansible Playbook
Josh

Ansible Architecture



Modules



- Modules are like tools
- Can be control system resources, like services, packages etc.
- 500+ core modules
- Also allows custom

INSTALLATION

--Commands to install Ansible on Master node

yum install https://dl.fedoraproject.org/pub/epel/epel-release-latest-7.noarch.rpm yum install ansible

--Configuration to setup SSH on Client and Master Machine

--Remove the comments from the ssh/sshd_config file which is to be overwritten

vi etc/ssh/sshd_config

-- After making the changes restart the sshd

systemctl restart sshd

--Generate the public key on Master machine which is to shared with clients

ssh-keygen

--following file will be created at id_rsa.pub which contains the value of public key

cd /root/.ssh

cat id_rsa.pub

--Copy the content of id_rsa.pub file

--Goto Client Machine and got to /root/.ssh folder and append the master's public key to authorized_keys files

--Add the hosts(IP addresses or in hosts file on the master machine under /etc/ansible folder

add client hosts

Ansible ADHOC COMMANDS

--**To find the documentation of the modules**

ansible-doc -l | more

ansible-doc -s yum

--To Run ping command for all hosts

ansible all -m ping

-**-To Run ping command to specific slave**

ansible -i hosts <<slave>> -m ping

--**To Run any command on the slave machine**

ansible -i hosts <<slave>> -m shell -a 'ls /home'

--Install A package on Client Machine

--I have added below clients int hosts file

172.31.27.53

172.31.19.90

172.31.31.119

YUM MODULE

--Lets install apache server on 172.31.27.53 server if it is already installed then yum remove httpd

--run the command in /etc/ansible folder because hosts file is available there

ansible 172.31.27.53 -m yum -a "name=httpd state=present"

--lets verify it on the slave machine (it should be available) service httpd status

SERVICE MODULE

--Now the service is present but it is not in started state on slave machine so lets Start the service on client using ansible on master machine

ansible 172.31.27.53 -m service -a "name=httpd state=started"

--Notice that State is changed=true

--Check the status on client machine (it should be in active state)

service httpd status

--Lets start service httpd again on master machine

ansible 172.31.27.53 -m service -a "name=httpd state=started"

--Notice that State is changed=false because it is already started

--Please try the same commands with state=stopped,restarted

--Repeat the steps to install nmap service

COPY MODULE

-- Create a file in master server and lets copy it to all or some of the slave servers

Step1 :-> Create a file in master machine (/tmp/testfile)

touch /tmp/testingfile

echo "test for ansible copy module" > /tmp/testfile

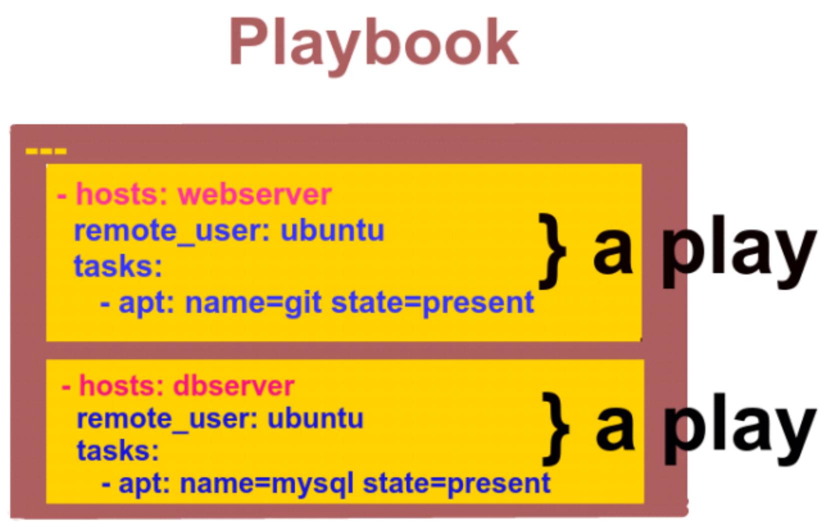
Step2:-> check on 172.31.27.53 that there is no file /tmp/testingfile

Step3:-> run the following command on master machine

ansible 172.31.27.53 -m copy -a "src=/tmp/testfile dest=/tmp/testfile"

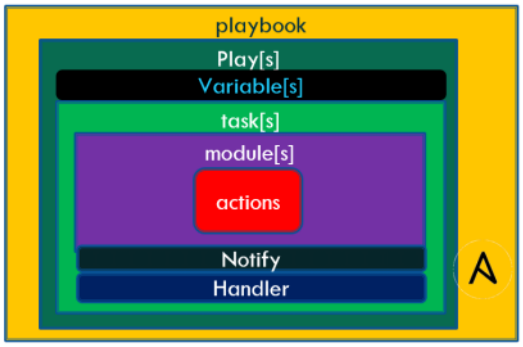
Step4:-> check on the slave machine it should have the file in /tmp/testfile

Ansible Play book



```
---
- hosts: webserver
  remote_user: ubuntu
  tasks:
    - apt: name=git state=present

- hosts: dbserver
  remote_user: ubuntu
  tasks:
    - apt: name=mysql state=present
```



- **Playbook** have number of **plays**
- **Play** contains Tasks
- **Task** calls core or custom **modules**
- **Task** can use **templates**
- **Handler** trigger from notify

executed at the end and only once

YAML(Ain't Markup Language) .yaml or .yml)

1. It is commonly used for configuration Management.
2.It is data serialization language designed to be directly writeable and readable by humans

3. Strictly speaking YAML is a superset of json with additional features like indentation or new line

4. It is a case sensitive scripting language.

Key

Key represents a variable or column for a value eg name: httpd in this case name is the key and httpd is the value

Data Types

Data Type represents the type of value we are storing in the key

eg x:25 # it is integer

x:"Ansible" # it is string

x:2.3 # "it is float

x:true # it is Boolean

x:null # it is null

Data Collection

Ans:-Generally when we represent the data it is key value pair which is called scalar representation of data.If we use multiple values or single key

or multiple key value pair it is called Data collection.

Data collection

It is of 2 types

Sequence data collection:- it is like array in other programming language

tasks:

- name:

- debug:

Map data collection:-it is equivalent to dictionary in python

tasks:

- name: home dir

- debug: true

Map data collection can have Sequence data collection

```
tasks:
- name: home dir
- debug:
var: result.stdout
```

Ansible Playbook

Ansible Playbook book is a yaml script. It sends the commands to remote server in scripted way instead of using Ansible commands indviually to configure remote server from command line.

Q:- What is Ansible Playbook Structure?

Ans:-Each playbook is an aggregation of one or more plays in it. Playbooks are structured using Plays. There can be more than one play inside a

playbook.The function of a play is to map a set of instructions defined against a particular host.

YAML is a strict typed language; so, extra care needs to be taken while writing the YAML files.

There are different YAML editors but we will prefer to use a simple editor like notepad++.

Just open notepad++ and copy and paste the below yaml and change the language to YAML (Language → YAML).

Egs

Task:-Install a apache server on remote machine

Play:-Consist of 1 or more tasks like install apache server and start the service

Playbook:-Composed or 1 or more play

----------Create a Playbook----------

A YAML starts with --- (3 hyphens)

Syntax

```
---
name: install and configure DB
hosts: testServer
vars:
oracle_db_port_value : 1521
tasks:
-name: Install the Oracle DB
yum: <code to install the DB>
-name: Ensure the installed service is enabled and running
service:
name: <your service name>
```

YAML tags

name

This tag specifies the name of the Ansible play. As in what this playbook will be doing. Any logical name can be given to the playbook.

hosts

This tag specifies the lists of hosts or host group against which we want to run the task. The hosts field/tag is mandatory. It tells Ansible on which hosts to run the listed tasks. The tasks can be run on the same machine or on a remote machine. One can run the tasks on multiple machines and hence hosts tag can have a group of hosts' entry as well.

vars

Vars tag lets you define the variables which you can use in your playbook. Usage is similar to variables in any programming language.

tasks

All playbooks should contain tasks or a list of tasks to be executed. Tasks are a list of actions one needs to perform. A tasks field contains the name of the task. This works as the help text for the user. It is not mandatory but proves useful in debugging the playbook. Each task internally links to a piece of code called a module. A module that should be executed, and arguments that are required for the module you want to execute.

What are the basic steps YAML script?

Step 1:- start with ---

Step 2:- Target section list(hosts,user, etc)

Step 3:- Variable list (optional)

Step 4:- Task list

List all the modules that you run, in the order

Step 5:- Save file with YAML

Assignment:-Run the following commands

on host1 and host2 :- execute sh file(date.sh)

on host1 :- find files and folders of /etc

on host2 :- run date command

on master:- find files and folders of /home

--------------------Variables--------------------------

What do you mean by Creating valid variable names?

Ans:Variable names should be letters, numbers, and underscores. Variables should always start with a letter.a1,a1_,file

Variable names should not be a reserved ansible keywords.

foo_port is a great variable. foo5 is fine too.

foo-port, foo port, foo.port and 12 are not valid variable names.

YAML also supports dictionaries which map keys to values. For instance:

foo:

field1: one

field2: two

You can then reference a specific field in the dictionary using either bracket notation or dot notation:

foo['field1']

foo.field1

Give a simple example of variables in YAML script

- hosts: 172.31.27.53

vars:

cmd1: sh /home/date.sh

name: play1

tasks:

- name: check current dir

command: "{{ cmd1 }}"

```
register: output
- debug:
var: output.stdout
```

#one more example

```
---
- hosts: 172.31.27.53
vars:
myvars: "This is my content"
tasks:
- copy:
dest: /tmp/var_file.txt
content: "{{ myvars }}"
- name: opening the file
command: cat /tmp/var_file.txt
register: output
- debug:
var: output.stdout
```

Q what is var_prompt

Ans

var_prompt is used to read the value for a variable at execution time

```
---
- name: This play book for var_prompt
hosts: 172.31.27.53
vars_prompt:
name: var1
prompt: Enter the value
tasks:
- name: output
debug:
msg: "This is the value of var1= {{ var1 }}"
```

#------------Run date command if /tmp/test.txt does not exist

```
---
- hosts: 172.31.27.53
vars:
test: "Testing"
tasks:
- name : Create file if not exist
command: date
register: output
args:
creates: /tmp/test.txt
- debug:
```

```
var: output.stdout
#-----------------------------------------------
```

#----------------------remove the file if exist

```
---
- hosts: 172.31.27.53
tasks:
- name: testing
command: 'touch /tmp/test'
args:
removes: /tmp/test
#------------------------------------------------
```

Conditional statement

```
# Determine if a path exists and is a directory. Note that we need to test
# both that p.stat.isdir actually exists, and also that it's set to true.
---
- hosts: 172.31.27.53
tasks:
- stat:
path: /tmp/test
register: p
- debug:
msg: "Path exists and is a directory {{ p }} "
when: p.stat.isdir is defined and p.stat.isdir
---
- hosts: 172.31.27.53
vars:
test: "True"
cont: "Hi from ansible"
tasks:
- copy:
dest: /tmp/test1.txt
content: "{{ cont }}"
when: ansible_facts['os_family'] == 'CentOs'
---
- hosts: 172.31.27.53
vars_prompt:
name: myvars
prompt: Enter the value
tasks:
- copy:
dest: /tmp/var_file.txt
content: "{{ myvars }}"
```

```
when: myvars == "test"
- name: opening the file
command: cat /tmp/var_file.txt
when: myvars == "test"
register: output
- debug:
var: output.stdout
when: myvars == "test"
---
- hosts: 172.31.27.53
vars_prompt:
name: myvars
prompt: Enter the value
tasks:
- copy:
dest: /tmp/var_file.txt
content: "{{ myvars }}"
when: myvars == "test"
- name: opening the file
command: cat /tmp/var_file.txt
ignore_errors: True
register: output
- debug:
var: output.stdout
when: myvars == "test"
---
- hosts: 172.31.27.53
vars_prompt:
name: myvars
prompt: Enter the value
name: bar
tasks:
- copy:
dest: /tmp/var_file.txt
content: "{{ myvars }}"
when: myvars == "test"
- name: opening the file
command: cat /tmp/var_file.txt
ignore_errors: True
register: output
- debug:
var: output.stdout
```

```
when: myvars == "test"
tasks:
- shell: echo "I've got and am not afraid to use it!"
when: foo is undefined
- fail: msg="Bailing out. this play requires 'bar'"
when: bar is undefined
```

multiple condtion

```
---
- hosts: 172.31.27.53
tasks:
- command: /tmp/test.sh
register: result
ignore_errors: True
- command: date
when: result is failed
- command: ls -l /home
when: result is succeeded
- command: ls -l /home/ec2-user
when: result is skipped
---
- hosts: 172.31.27.53
vars:
var1: 1
var2: 2
tasks:
- name: This is for condition1
command: date
when: var1 == 1 or var2 ==2
- name: This is for condition2
command: date
when:
- var1 == 1
- var2 == 2
```

Loops

Q:-What is loop in YAML?

Ans:- It is the repeation of tasks

example:-If you want to create 3 directories on host machine

```
- hosts: 172.31.27.53
tasks:
- name: Create a dir1
command: mkdir /tmp/dir1
-name: Create a dir2
```

```
command: mkdir /tmp/dir2
-name: Create dir3
command: mkdir /tmp/dir3
Now lets create above tasks with Loops
---
- hosts: 172.31.27.53
tasks:
- name: Create a dire
command: mkdir /tmp/"{{ item }}"
with_items:
- new_dir1
- new_dir2
- new_dir3
----------Create users in host machine
---
- hosts: 172.31.27.53
tasks:
- name: add several users
user:
name: "{{ item }}"
state: present
groups: "wheel"
with_items:
- testuser1
- testuser2
```

Ansible-Vault

Ansible Vault is a feature of ansible that allows you to keep sensitive data such as passwords or keys in encrypted files,
rather than as plaintext in playbooks or roles. These vault files can then be distributed or placed in source control.

```
ansible-vault create t.yaml  # To create encrypted files, it will ask password for encryption
cat t.yaml  # it will show encrypted data
ansible-vault edit t.yaml  # To edit the vault file
ansible-vault decrypt t.yaml # To decrypt the file
ansible-vault encrypt t.yaml # To encrypt the file
```

Ansible-Roles

--Ansible Roles
Roles in Ansible are next level of abstraction of Ansible playbooks
--Benefits of Ansible Roles
  idea of include files and combine them to form clean and resuable abstraction
  Easy to maintain/troubleshooting the playbooks

--Structure of Roles

files: contains the regular files those need to copy to target folder
handlers: Event handlers
meta: Role dependencies
templates: similar to files but contains dynamic data
tasks: playbook tasks
vars/group_vars: variable definitions

```
ansible-galaxy search apache
galaxy.ansible.combine

ansible-galaxy init apache --offline
```

main.yml

```
---
# tasks file for apache
- include: install.yml
- include: configure.yml
- include: service.yml
```

```
#install.yml

---
 name: installing httpd
 yum:
  name: httpd
  state: present
```

```
#configure.yml
---
- name: httpd conf
  copy: src=httpd.conf dest=/etc/httpd/conf/httpd.conf
  notify:
     restart apache service
- name: send the file
  copy: src=index.html dest=/var/www/html/index.html
```

```
# copy the configuration file to files/ folder
cp /etc/httpd/conf/httpd.conf .
```

# under handlers folder/main.yml

```
---
# handlers file for apache
- name: restart apache service
  service: name=httpd state=restarted
```