Vijay's Blog

Follow

Fundamentals of Ansible

vijaykumar guntireddy

Aug 26, 2024 · 12 min read

Table of contents

Configuration Management

- Converting a plain server to ready serve a application

Disadvantages of Shell scripting

- Not Idempotent
- Error handling
- Homogenous
- not scalable when too many servers
- syntax is not easy to understand

CM tools - Ansible , puppet , chef , rundeck etc.

We have two types of architecture in CM

- push vs pull

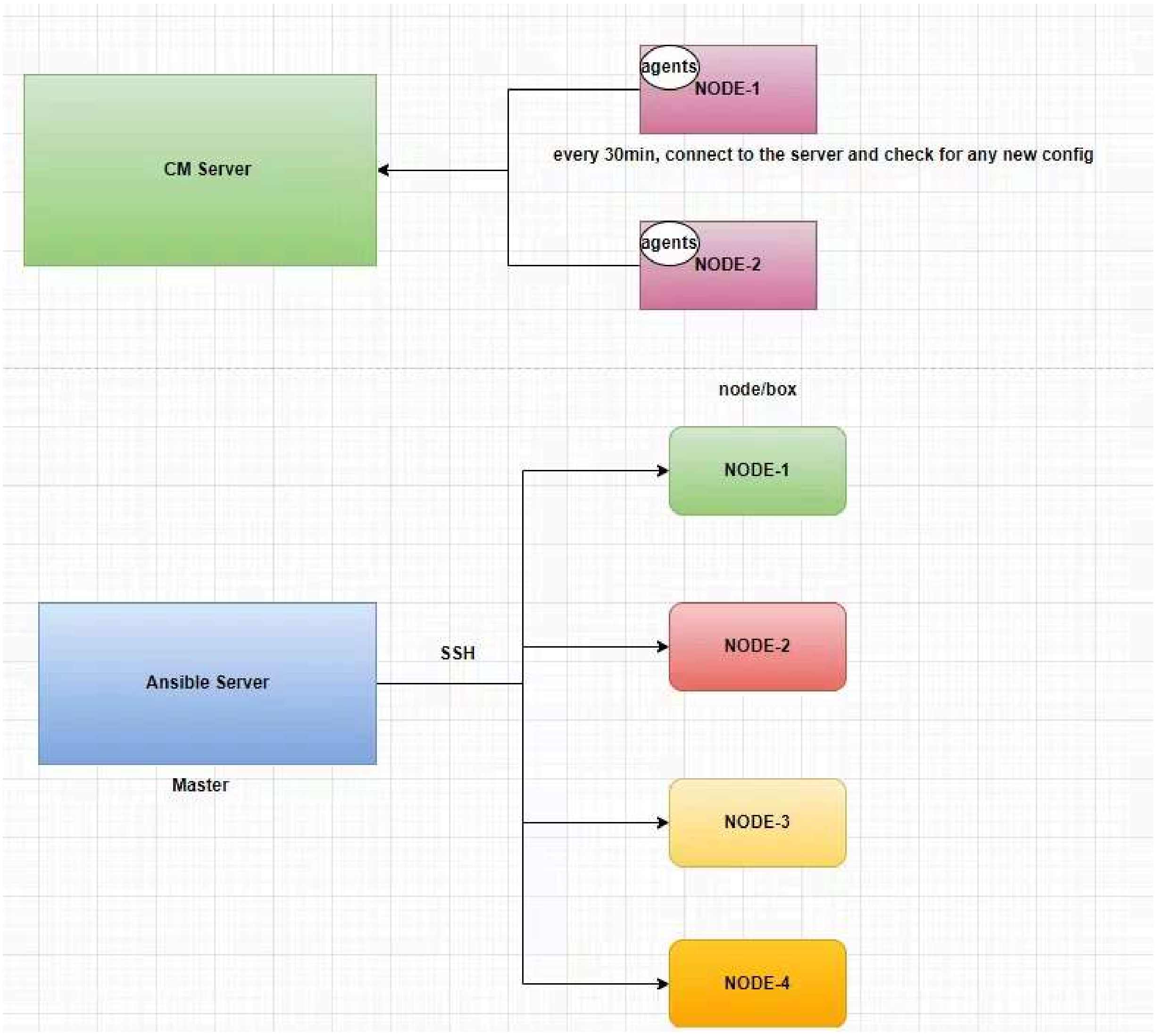
Lets understand this with simple example

- we are getting a courier from Delhi to Hyderabad.
- we have two options
    - Go to the courier office everyday and check for courier - **PULL**

    - Wait in the home and courier office will deliver when every package is received. - **PUSH**
- Disadvantages in PULL
    - more traffic in internet
    - bandwidth
    - cost

Ansible

- **Ansible is push based architecture**



- Ansible will connect to all nodes(servers) and loads the configuration

- Ansible uses SSH protocol.
- In Ansible 10% cases it uses PULL.
- The system where ansible is loaded is called ansible server and all nodes will be connected to this ansible server.

Ansible Inventory

- List of servers ansible is managing
- Linux commands - dnf etc → these are called as modules in ansible.
- Do following operations
  - Connect to node
  - install nginx
  - run ngnix

COPY

```
#From ansible server
$ansible -i 172.31.41.249, all -e ansilbe_user=ec2-user -e ansible_
172.31.41.249 | UNREACHABLE! => {
    "changed": false,
    "msg": "Failed to connect to the host via ssh: ssh: connect to
    "unreachable": true
}

#install it as a root option -b
$ansible -i 18.209.110.154, all -e ansilbe_user=ec2-user -e ansible
18.209.110.154 | CHANGED => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python3"
    },
    "changed": true,
    "msg": "",
    "rc": 0,
    "results": [
        "Installed: nginx-filesystem-1:1.20.1-14.el9_2.1.noarch",
        "Installed: nginx-core-1:1.20.1-14.el9_2.1.x86_64",
        "Installed: redhat-logos-httpd-90.4-2.el9.noarch",
        "Installed: nginx-1:1.20.1-14.el9_2.1.x86_64"
    ]
}

#if already installed state will not change
$ ansible -i 18.209.110.154, all -e ansilbe_user=ec2-user -e ansibl
18.209.110.154 | SUCCESS => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python3"
    },
    "changed": false,
    "msg": "Nothing to do",
    "rc": 0,
    "results": []
}

#To start ngnix
$ansible -i 18.209.110.154, all -e ansilbe_user=ec2-user -e ansible
```

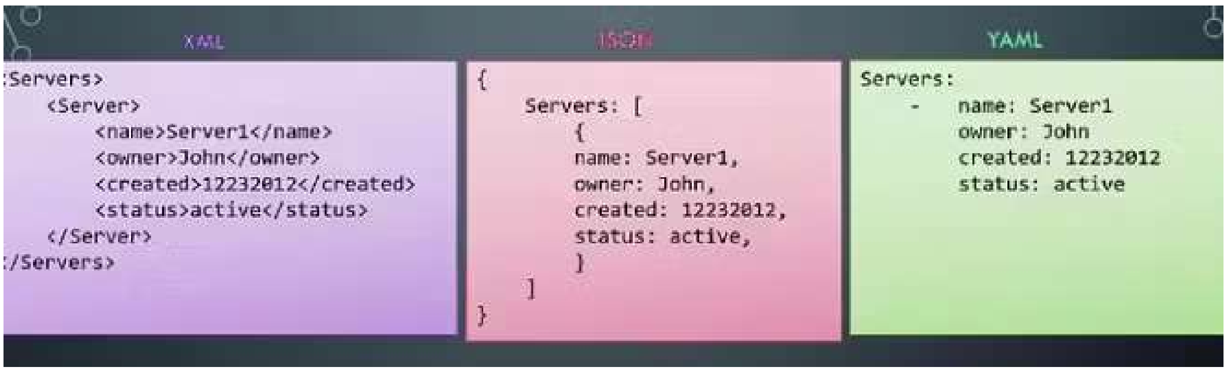
Adhoc commands

- command issued from ansible server targeting node manually, basically on some emergency/adhoc purpose

Running these commands in one file with simple syntax can be done with Ansible

Playbooks (yet another markup language)

- list of plays which contains modules that can do specific task in the ansible server.



```
<Servers>
    <Server>
        <name>Server1</name>
        <owner>John</owner>
        <created>12232012</created>
        <status>active</status>
    </Server>
</Servers>
```

```
{
    Servers: [
        {
            name: Server1,
            owner: John,
            created: 12232012,
            status: active,
        }
    ]
}
```

```
Servers:
    -   name: Server1
        owner: John
        created: 12232012
        status: active
```

- Ansible modules documentation : https://docs.ansible.com/ansible/2.9/modules/list_of_all_modules.html

Sample Playbook which will do Ping operation

created 01-ping.yaml file with below content

COPY

```
- name: ping operation
  host: web #which hosts ansible can connect to
  task: # list of modules
  - name: ping the server
    ansible.builtin.ping:
```

Inventory - List of servers where we can group .

created inventory.ini file with IP address of the node

COPY

```
[web]
172.31.35.211
```

add this folder to your github repository

COPY

```
vijay@vijay_lenovo MINGW64 /c/Vijay/devops (master)
$git remote add origin <github url>
$git add . ; git commit -m "ansible" ; git push origin master
```

download these files in AWS ansible server and execute ansible command

The ansible node will be pinged from Ansible server.

COPY

```
#install ansible
$ sudo dnf install ansible -y
#download the project
$ git clone https://github.com/qtivijay/devopsaws.git
$ cd devopsaws/
$ cd ansible1/
#execute ansible command
$ ansible-playbook -i inventroy.ini -e ansible_user=ec2-user -e ans

PLAY [ping the server] *******************************************

TASK [Gathering Facts] *******************************************
ok: [172.31.35.211]

TASK [ping the server] *******************************************
ok: [172.31.35.211]

PLAY RECAP *******************************************************
172.31.35.211              : ok=2    changed=0    unreachable=0
```

Multiplay example

COPY

```
- name: multiplay
  hosts: web
  tasks:
  - name: play01 task01
    ansible.builtin.debug:
      msg: "This is from PLAY-01 and TASK-01"

- name: multiplay
  hosts: web
  tasks:
  - name: play02 task01
    ansible.builtin.debug:
      msg: "This is from PLAY-02 and TASK-01"
```

Install and run nginx (imp: **become**)

COPY

```
- name: nginx install and run
  hosts: web
  become: yes #take sudo access
  tasks:
  - name: install nginx
    ansible.builtin.package: #heterongenous module that can work fc
      name: nginx
      state: present

  - name: run nginx
    ansible.builtin.service:
      name: nginx
```

```
        state: started
        enabled: yes
```

Ansible Variables

1 . Declaring variables in yaml (**vars**)

COPY

```
- name: variables to print
  hosts: web
  vars:
    COURSE: "DevOps with AWS"
    DURATION: "120HRS"
    TRAINER: "Sivakumar Reddy M"
  tasks:
  - name: print the information
    ansible.builtin.debug:
      msg: "Hi, I am learning {{COURSE}}, Duration is: {{DURATION}
```

2. Task level variables (**vars**)

COPY

```
- name: variables to print
  hosts: web
  vars: #play level, all tasks in this play have access to this
    COURSE: "DevOps with AWS"
    DURATION: "120HRS"
    TRAINER: "Sivakumar Reddy M"
  tasks:
  - name: print the information
    ansible.builtin.debug:
      msg: "Hi, I am learning {{COURSE}}, Duration is: {{DURATION}
  - name: print the information again
    vars: #task level variables override play level
      COURSE: "Ansible"
    ansible.builtin.debug:
      msg: "Hi, I am learning {{COURSE}}, Duration is: {{DURATION}}

  - name: print the information again and again
    ansible.builtin.debug:
      msg: "Hi, I am learning {{COURSE}}, Duration is: {{DURATION}}
```

3. variables in another file (**vars_file**)

COPY

```
- name: variables from files
  hosts: web
  vars_files:
  - vars.yaml
  tasks:
  - name: print the course information
    ansible.builtin.debug:
      msg: "Hi, I am learning {{COURSE}}, Duration is: {{DURATION}}
```

vars.yaml

COPY

```
COURSE: "DevOps with AWS from Files"
DURATION: "150HRS"
TRAINER: "Sivakumar Reddy M"
```

4. Variables through prompt (**vars_prompt , prompt , private**)

COPY

```
- name: variables from prompt
  hosts: web
  vars_prompt:
  - name: COURSE
    prompt: Please enter the course name
    private: false # not confidential
  - name: TRAINER
    prompt: Please enter the trainer name
    private: false # not confidential
  - name: DURATION
    prompt: Please enter the duration
    private: false # not confidential
  tasks:
  - name: print the course information
    ansible.builtin.debug:
      msg: "Hi, I am learning {{COURSE}}, Duration is: {{DURATION}}
```

5. Variables from inventory file (**[web:vars]**)

COPY

```
[web]
172.31.43.207

[web:vars]
COURSE="AWS"
DURATION="10HRS"
TRAINER="Sivakumar Reddy"
```

COPY

```
- name: variables to print
  hosts: web
  tasks:
    - name: print the information
      ansible.builtin.debug:
        msg: "Hi, I am learning {{COURSE}}, Duration is: {{DURATION}}
```

6. Variables from command arguments (**-e "args"**)

COPY

```
- name: variables to print
  hosts: web
  tasks:
  - name: print the information
    ansible.builtin.debug:
      msg: "Hello {{NAME}}, Good {{GREETING}}"
```

COPY

```
command : ansible-playbook -i inventroy.ini -e ansible_user=ec2-use
```

Ansible variables preference order

COPY

```
#1. command line/args
#2. Task level - vars
#3. Files - vars_files:
#4. prompt - vars_prompt
#5. Play - vars
#6. inventory - [web:vars]
```

COPY

```
- name: variables to print
  hosts: web
  # vars:
  #   COURSE: "DevOps with AWS from PLAY level"
  #   DURATION: "120HRS"
  #   TRAINER: "Sivakumar Reddy M"
  # vars_files:
  # - vars.yaml
  # vars_prompt:
  # - name: COURSE
  #   prompt: Please enter the course name
  #   private: false # not confidential
  tasks:
  - name: print the information
    vars:
      #COURSE: "DevOps with AWS from TASK level"
      DURATION: "120HRS"
      TRAINER: "Sivakumar Reddy M"
    ansible.builtin.debug:
      msg: "Hi, I am learning {{COURSE}}"
```

Ansible inventory

In Ansible inventory we have three types

- ungrouped
- grouped
- group of groups

```
192.168.1.1
192.168.1.2
192.168.1.3
```

```
[web]
172.31.45.75

[web:vars]
DURATION="10HRS"
TRAINER="Sivakumar Reddy"

[backend]
192.168.1.4
192.168.1.5
192.168.1.6

[mysql]
192.168.1.7
192.168.1.8
192.168.1.9

[servers:children]
web
backend

[local]
localhost

[web:vars]
ansible_user=ec2-user
ansible_password=DevOps321
```

COPY

```
# ansible -i inventroy.ini ungrouped --list-hosts
  hosts (3):
    192.168.1.1
    192.168.1.2
    192.168.1.3
# ansible -i inventroy.ini mysql --list-hosts
```

```
  hosts (3):
    192.168.1.7
    192.168.1.8
    192.168.1.9



# ansible -i inventroy.ini servers --list-hosts
  hosts (4):
    172.31.45.75
    192.168.1.4
    192.168.1.5
    192.168.1.6

# ansible -i inventroy.ini all --list-hosts
  hosts (11):
    192.168.1.1
    192.168.1.2
    192.168.1.3
    192.168.1.7
    192.168.1.8
    192.168.1.9
    localhost
    172.31.45.75
    192.168.1.4
    192.168.1.5
    192.168.1.6
```

Ansible Datatypes

Strings,numbers,maps,lists,boolean

connection: local → ansible will not ask for username and password

COPY

```
- name: devops course information
  hosts: local
  connection: local # for localhosts
  vars:
    course: "DevOps with AWS" #string
    duration: 120 #number
    topics: #list
    - Linux
    - Shell
    - Ansible
    - Terraform
    - AWS
    - K8
    live: true #boolean
    tools: #map/dictionary
      ci: jenkins # string
      cm: ansible
      cloud: aws
      aws:
      - IAM
      - S3
  tasks:
  - name: print the course information
    ansible.builtin.debug:
      msg: "Hi, I am learning {{ course }}, with duration: {{ durat
```

Below command

- automatically picks the localhost (inventory.ini) for that username and password is not required

COPY

```
ansible-playbook -i inventroy.ini 10-datatypes.yaml
```

Ansible conditions

Ansible "When" is used to decide whether particular task/module run or not .

https://docs.ansible.com/ansible/latest/playbook_guide/playbooks_conditionals.html

COPY

```
- name: check a number is less than 10 or not
  hosts: local
  connection: local
  vars_prompt:
  - name: my_number
    prompt: Please enter the number
    private: false # not confidential
  # vars:
  #   my_number: 139 # number
  tasks:
  - name: print this if less than 10
    ansible.builtin.debug:
      msg: "Given number {{ my_number }} is less than 10"
    when: my_number | int < 10 # conditional expression

  - name: print this if greater than or equal to 10
    ansible.builtin.debug:
      msg: "Given number {{ my_number }} greater than or equal to 1
    when: my_number | int >= 10
```

Lets understand with another example

Task is to create a new user

- Check already user exists or not
- if exists don't create
- if doesn't exists create

For above tasks ansible doesn't have modules then we will use commands

COPY

```
- name: create user
  hosts: web
  become: yes
  tasks:
  - name: check user exist or not
    ansible.builtin.command: id expense
    register: USER #here USER is a variable that gets output from a
    ignore_errors: True

  - name: print the user information
    ansible.builtin.debug:
      msg: "user info: {{ USER }}"

  - name: create user
    ansible.builtin.command: useradd expense
    when: USER.rc != 0
```

Things to watch out in above file

- "ignore_errors : True" →Ansible will not stop if the task failed, it will ignore and proceed with next command

- "register"⟶ helps to store output from the command
- For "id username" , for below output "**rc : 1**" means no such user (based on this yaml conditon can be written)

COPY

```
 "user info: {
'changed': True,
'stdout': '',
'stderr': 'id: ‘expense’: no such user',
'rc': 1,
'cmd': ['id', 'expense'],
'start': '2024-09-04 07:36:02.795263',
'end': '2024-09-04 07:36:02.807144',
'delta': '0:00:00.011881',
'failed': True,
'msg': 'non-zero return code',
'stdout_lines': [],
'stderr_lines': ['id: ‘expense’: no such user']}"
```

- If user already there then we will get following output

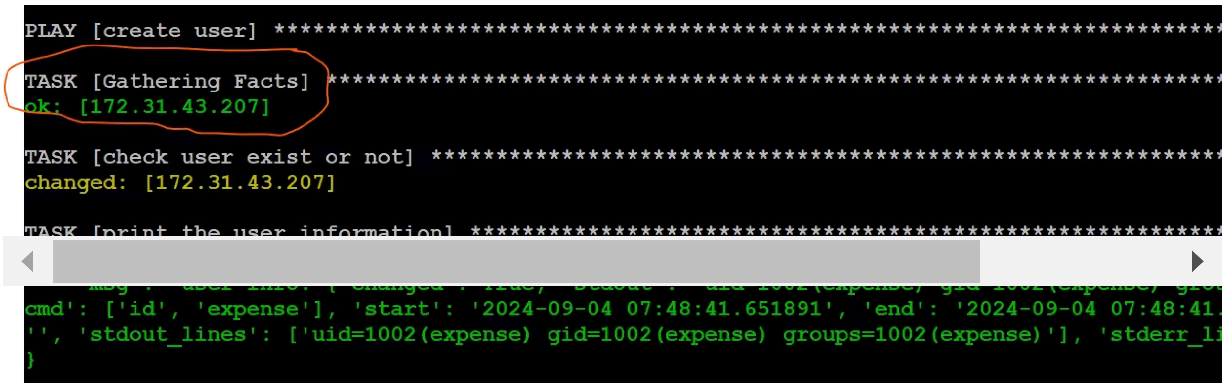
COPY

```
 "user info:
{'changed': True,
'stdout': 'uid=1002(expense) gid=1002(expense) groups=1002(expense)
'stderr': '',
'rc': 0,
'cmd': ['id', 'expense'],
'start': '2024-09-04 07:48:41.651891',
'end': '2024-09-04 07:48:41.659323',
'delta': '0:00:00.007432',
'msg': '',
    'stdout_lines': ['uid=1002(expense) gid=1002(expense) groups=1002(e
    'stderr_lines': [], 'failed': False}
```

Ansible Gathering Facts

Facts and variables both are same

Ansible, before connections to the servers/hosts it will collect entire information. so that it can take decisions based on that information



```
PLAY [create user] ************************************************************

TASK [Gathering Facts] ********************************************************
ok: [172.31.43.207]

TASK [check user exist or not] ************************************************
changed: [172.31.43.207]

TASK [print the user information] *********************************************

cmd': ['id', 'expense'], 'start': '2024-09-04 07:48:41.651891', 'end': '2024-09-04 07:48:41.
'', 'stdout_lines': ['uid=1002(expense) gid=1002(expense) groups=1002(expense)'], 'stderr_li
}
```

Lets understand "Gathering Facts" with an example

- we already have below package information for loading nginx in redhat
    - redhat → ansible.builtin.dnf
- Now we have to use same ansible to support loading nginx in ubuntu
    - ubuntu → ansible.builtin.apt
- Lets see how we can resolve this ..?

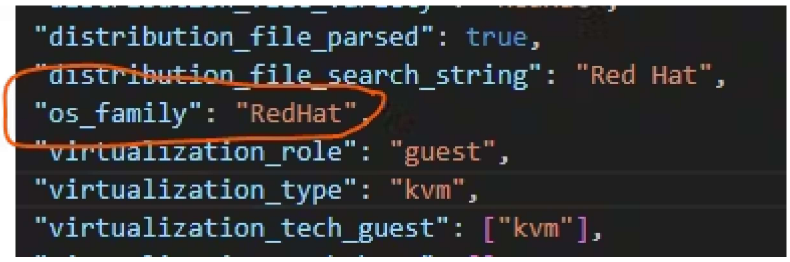
COPY

```
- name: gathering facts
  hosts: web
  become: yes
  tasks:
  - name: print the facts
    ansible.builtin.debug:
      msg: " {{ ansible_facts }} "
```

"ansible_facts" varible is there in ansible which is used to print the "gathering facts"

after beautifying the output is
https://github.com/qtivijay/devopsaws/blob/master/ansible1/facts.json

we will use below variable from `ansible_facts` json to know whether it is ubuntu or redhat



```
"distribution_file_parsed": true,
"distribution_file_search_string": "Red Hat",
"os_family": "RedHat",
"virtualization_role": "guest",
"virtualization_type": "kvm",
"virtualization_tech_guest": ["kvm"],
```

```
- name: install nginx
  become: yes
  hosts: web
  tasks:
  - name: print the stats
    ansible.builtin.debug:
      msg: "{{ ansible_facts }}"

  - name: install nginx if redhat
    ansible.builtin.dnf:
      name: nginx
      state: present
      when: ansible_os_family == "RedHat"

    - name: install nginx if Debian
      ansible.builtin.apt:
        name: nginx
        state: present
      when: ansible_os_family == "Debian"
```

The above yaml will install the nginx based on os.

Ansible loops

COPY

```
- name: loops demo
  hosts: local
  connection: local
  tasks:
  - name: print names
    ansible.builtin.debug:
      msg: "Hello {{ item }} "
    loop:
    - Ramesh
    - Suresh
    - Robert
    - Raheem
```

item is a reserved ansible keyword to parse the loop

COPY

```
- name: loops demo
  hosts: web
  become: yes
  tasks:
```

```
  - name: install packages
    ansible.builtin.package:
      name: "{{ item }}"
      state: present
    loop:
    - mysql
    - nginx
    - postfix
    - httpd
```

COPY

```
- name: loops demo
  hosts: web
  become: yes
  tasks:
  - name: install packages
    ansible.builtin.package:
      name: "{{ item.name }}"
      state: "{{ item.state }}"
    loop:
    - { name: 'mysql', state: 'present' }
    - { name: 'nginx', state: 'absent' }
    - { name: 'postfix', state: 'absent' }
    - { name: 'nginx', state: 'present' }
```

Ansible Functions/Filters

- Ansible allow user to write custom functions (using python language)
- But 90% of cases we use built-in ansible functions in project.
- https://docs.ansible.com/ansible/latest/playbook_guide/playbooks_filters.html

- Ansible filters are used to manipulate data within your playbooks. (Pipe |)

COPY

```
- name: demo on filters/functions
  hosts: local
  connection: local
  tasks:
  - name: print the default variable
    ansible.builtin.debug:
      msg: "Hello {{ course | default('Ansible') }}"

  - name: convert to uppercase
    vars:
      greeting: "Hello Ramesh"
    ansible.builtin.debug:
      msg: " {{ greeting | upper }}"

  - name: convert to lowercase
    vars:
      greeting: "Hello Ramesh"
    ansible.builtin.debug:
      msg: " {{ greeting | lower }}"

  - name: get the unique values
    vars:
      numbers: [1,2,3,45,4,3,2]
    ansible.builtin.debug:
      msg: " {{ numbers | unique }}"

  - name: get the min and max
    vars:
      numbers: [1,2,3,45,4,3,2]
    ansible.builtin.debug:
      msg: " Min: {{ numbers | min }}, Max: {{ numbers | max }}"
```

```yaml
- name: convert map to list
  vars:
    course:
      name: "DevOps with AWS"
      duration: 120
      trainer: "sivakumar reddy"
  ansible.builtin.debug:
    msg: "{{ course | dict2items }}"

- name: convert list to map
  vars:
    course:
    - { "key": "name","value": "DevOps with AWS" }
    - { "key": "duration","value": 120 }
    - { "key": "trainer","value": "sivakumar reddy" }
  ansible.builtin.debug:
    msg: "{{ course | items2dict }}"

- name: check ip address
  vars:
    myip: "356.168.1.1"
  ansible.builtin.debug:
    msg: "{{ myip | ansible.utils.ipv4 }}"
```

Command vs Shell

- **Use** `command` when you need to run simple commands that do not require shell features. It is more secure and should be preferred for basic tasks.
- **Use** `shell` when you need to leverage shell features like pipes, redirection, or environment variables. It provides more flexibility but should be used with caution due to potential security risks

ansible.builtin.command , ansible.builtin.shell

COPY

```
- name: check the process
    ansible.builtin.shell: ps -ef | grep ssh
    register: output

  - name: check the process
    ansible.builtin.command: ps -ef | grep ssh  # will fail as pipe
    register: output
```

Challenges and solutions

1. Downloaded the code from the github and made some changes and again i want to revert to latest changes in github , how to do it ..?

COPY

```
git reset --hard origin/master
```

2. Adding file to staging area to commit and to github in oneline ..?

COPY

```
git add . ; git commit -m "ansible" ; git push origin master
```

Possible interview questions

1. Write ansible code to download and run nginx ..?

    1. https://github.com/qtivijay/devopsaws/blob/master/ansible1/03-nginx.yaml
2. What is difference between command vs shell in ansible ..?
    1. check command vs shell section

ansible

Written by

vijaykumar guntireddy

Results-oriented Global Management Professional with diverse experience of >19 years in different capacities primarily in end-to-end product management, quality, delivery of critical business applications, and project management.

Follow

MORE ARTICLES

vijaykumar guntireddy