Ansible is simple, open-source, configuration management tool used for IT automation engine for cloud infrastructure, in-house servers.

Ansible push based configuration management system/tool

1. ansible does not require any dedicated agent running on the target host machines.
2. minimum ansible requirement is host machines with python installed on it.
3. we only require a proper ssh connection between the controller and the host machines.

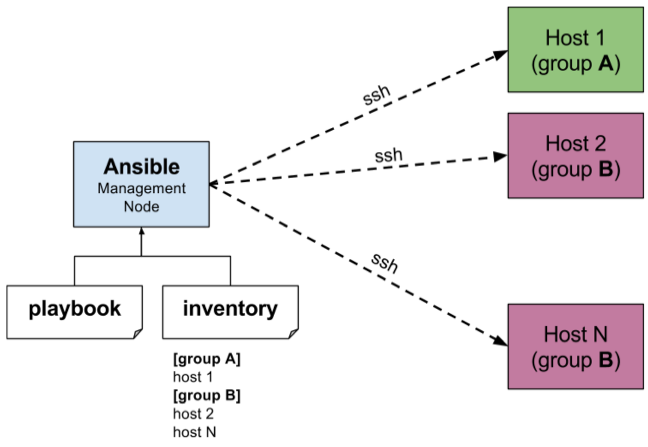
Advantages:

1. free/open source
2. Very simple to setup
3. Agent less CM tool

playbook

- can contain n number of play
- each play is designated to run n number task
- each task is designated to execute a module (only one module per task)

Ansible architecture



Ansible configuration

1. ANSIBLE_CONFIG (environment variable if set)
2. ansible.cfg (in the current directory)
3. ~/.ansible.cfg (in the home directory)
4. /etc/ansible/ansible.cfg

Ansible Inventory Static Inventory

- It is file which contains the ip and configuration of connection to target host machines on which
  we want execute our playbooks
- we can group the hosts in inventory.
- default location of host file is /etc/ansible/hosts
- each line except group name is considered as a single host connection configuration.

```
 [frontend]
13.233.104.27 ansible_user=ubuntu ansible_become_user=root
15.206.185.128 ansible_user=ubuntu ansible_become_user=root

 [backend]
15.206.185.128 ansible_user=ubuntu ansible_become_user=root
```

Dynamic Inventory

- Is basically a script which will gives the inventory output.
- we can use python, perl, bash

Ansible playbook

```
---
 - name: Play_2
   hosts: backend
   tasks:
     - name: Install git (backend)
       become: true
       apt:
         name: git
         state: present
         update_cache: yes
     - name: Install jq (backend)
       become: true
       apt:
```

name: jq

state: latest

update_cache: yes

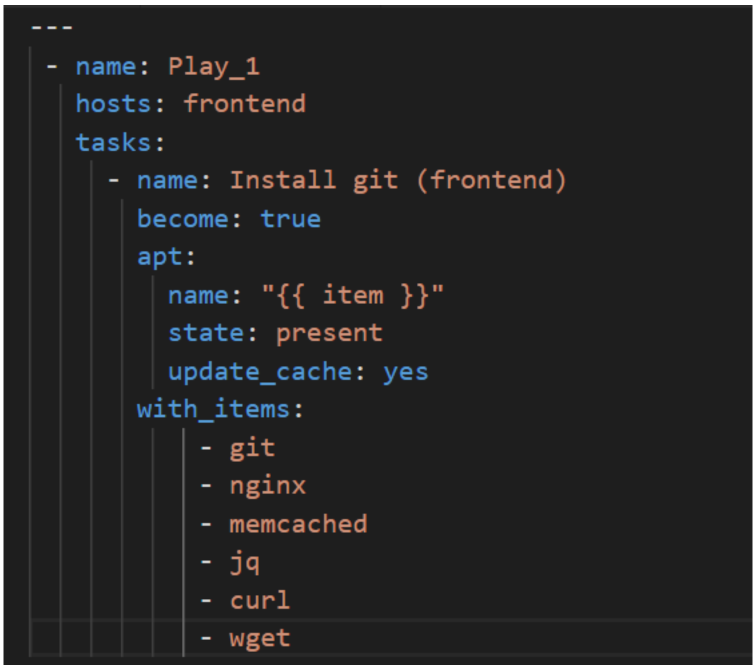
commands

$ ansible all -m ping

$ ansible-playbook -k playbook1.yaml

$ ansible-playbook -k playbook1.yaml -i <inventory file>

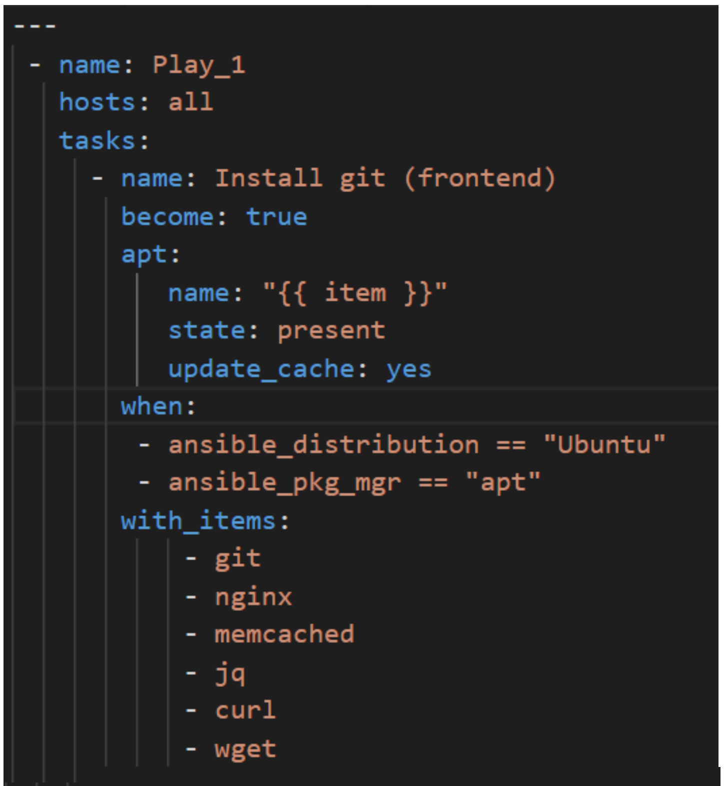
Installing multiple items with single task



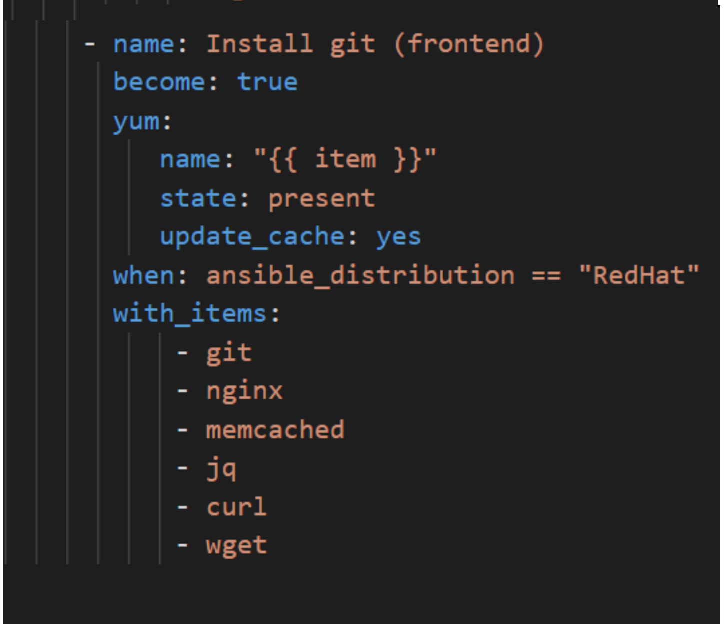
```yaml
---
- name: Play_1
  hosts: frontend
  tasks:
    - name: Install git (frontend)
      become: true
      apt:
        name: "{{ item }}"
        state: present
        update_cache: yes
      with_items:
          - git
          - nginx
          - memcached
          - jq
          - curl
          - wget
```

Ansible facts

- Ansible facts are the variable which contains the complete details of hosts
- all the facts are prefixed with ansible_ word
- setup module is used to gather all facts (ansible all -m setup)



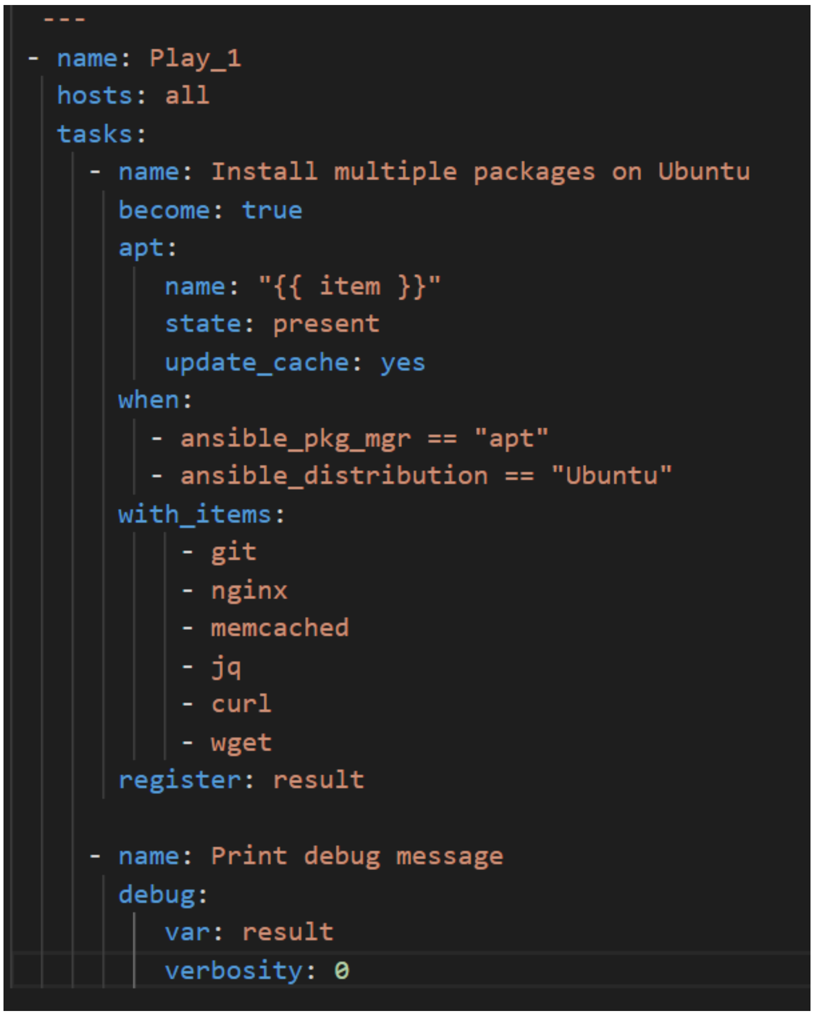
```yaml
---
 - name: Play_1
   hosts: all
   tasks:
     - name: Install git (frontend)
       become: true
       apt:
          name: "{{ item }}"
          state: present
          update_cache: yes
       when:
        - ansible_distribution == "Ubuntu"
        - ansible_pkg_mgr == "apt"
       with_items:
           - git
           - nginx
           - memcached
           - jq
           - curl
           - wget
```



```yaml
     - name: Install git (frontend)
       become: true
       yum:
          name: "{{ item }}"
          state: present
          update_cache: yes
       when: ansible_distribution == "RedHat"
       with_items:
           - git
           - nginx
           - memcached
           - jq
           - curl
           - wget
```

Ansible debug

- print output of a task with register variable

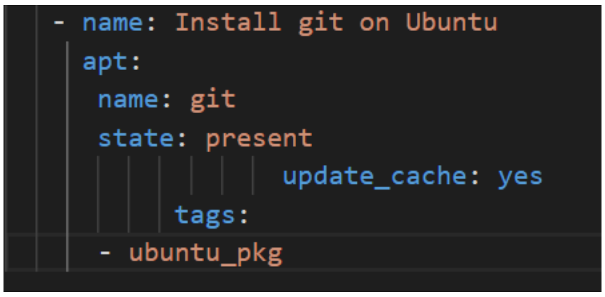


```
---
- name: Play_1
  hosts: all
  tasks:
    - name: Install multiple packages on Ubuntu
      become: true
      apt:
        name: "{{ item }}"
        state: present
        update_cache: yes
      when:
        - ansible_pkg_mgr == "apt"
        - ansible_distribution == "Ubuntu"
      with_items:
          - git
          - nginx
          - memcached
          - jq
          - curl
          - wget
      register: result

    - name: Print debug message
      debug:
        var: result
        verbosity: 0
```

Ansible Run specific tasks / plays

- Using tags we can run specific task or play in a playbook

$ ansible-playbook playbook.yaml --tags="ubuntu_pkg"



```
  - name: Install git on Ubuntu
    apt:
     name: git
     state: present
               update_cache: yes
          tags:
   - ubuntu_pkg
```

Ansible Roles

- Roles are better way of organizing my playbooks
- Instead of defining everything play, handlers, templates etc. in a single file
  we can define it in a proper folder structure using roles
- default location od roles is /etc/ansible/role (if directory is not present create it)
- ansible-galaxy is the tool which ansible gives us to create a role

ansible-galaxy init <role_name>

- Role directory structure

```
roles
    └── install_pkg
            ├── README.md
            ├── defaults
            │   └── main.yml
            ├── files
            ├── handlers
            │   └── main.yml
            ├── meta
            │   └── main.yml
            ├── tasks
            │   └── main.yml
            ├── templates
            ├── tests
            │   ├── inventory
            │   └── test.yml
            └── vars
                    └── main.yml
```

tasks - Contains the main list of task that we want to execute by the role.
main.yml is where we put our tasks

handlers - Contains tasks which are executed only if it is notified by other task after
successful execution (if there is change from the notifying task).

defaults - It contains default variables for the role and it is the least precedence

variable in ansible.

vars - The variables which will be used by role. These variable can be defined in our

playbook also. (task/play variables)

files - Contains static file that can be copied to hosts by role

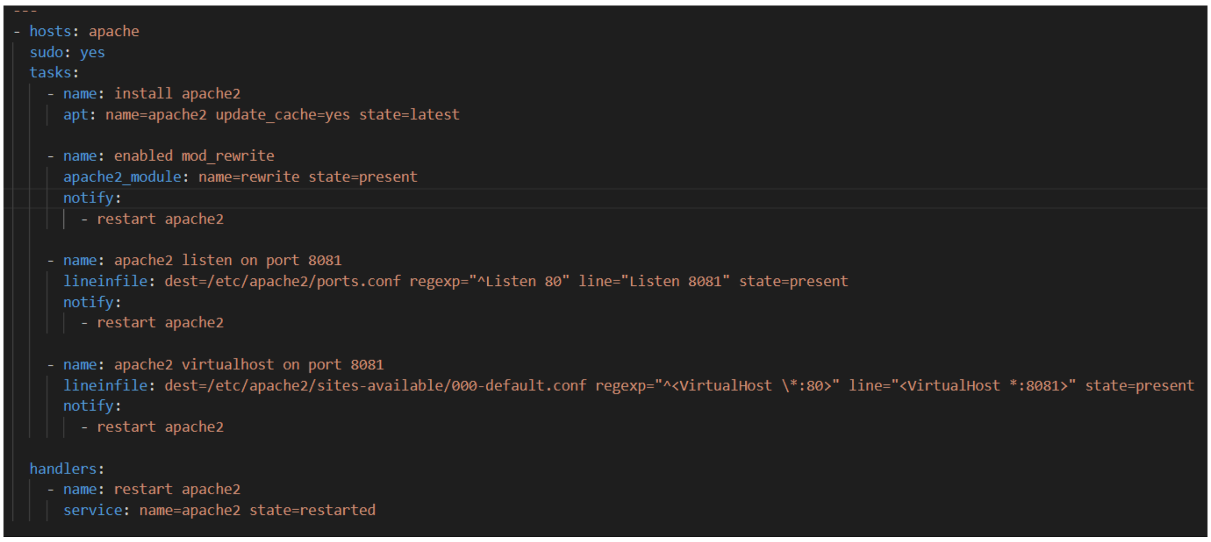
In task I don't need use the complete path just filename is used to copy.

templates - used to define dynamic value in files and copy it to host with values defined.

this uses jinja2 format templating.

meta - To define role dependencies.

Single playbook



```
---
- hosts: apache
  sudo: yes
  tasks:
    - name: install apache2
      apt: name=apache2 update_cache=yes state=latest

    - name: enabled mod_rewrite
      apache2_module: name=rewrite state=present
      notify:
        - restart apache2

    - name: apache2 listen on port 8081
      lineinfile: dest=/etc/apache2/ports.conf regexp="^Listen 80" line="Listen 8081" state=present
      notify:
        - restart apache2

    - name: apache2 virtualhost on port 8081
      lineinfile: dest=/etc/apache2/sites-available/000-default.conf regexp="^<VirtualHost \*:80>" line="<VirtualHost *:8081>" state=present
      notify:
        - restart apache2

  handlers:
    - name: restart apache2
      service: name=apache2 state=restarted
```

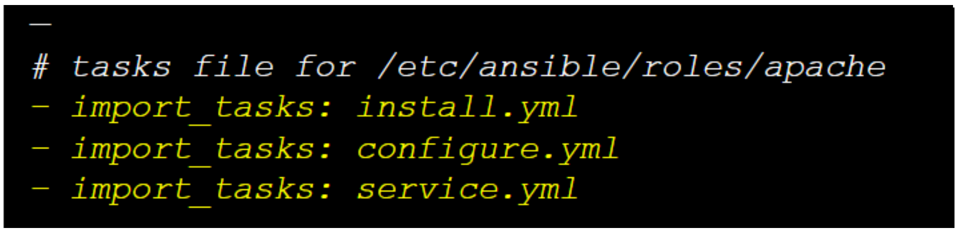
Create a Role

```
# ansible-galaxy init /etc/ansible/roles/apache –offline
```

Distribute the tasks into the role

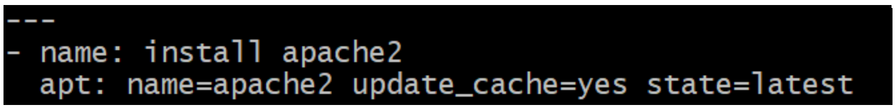
1. Let's create install.yml, confgure.yml, service.yml included in the main.yml with actions in the same directory.

```
$ cd /etc/ansible/roles/apache/
$ cat tasks/main.yml
```



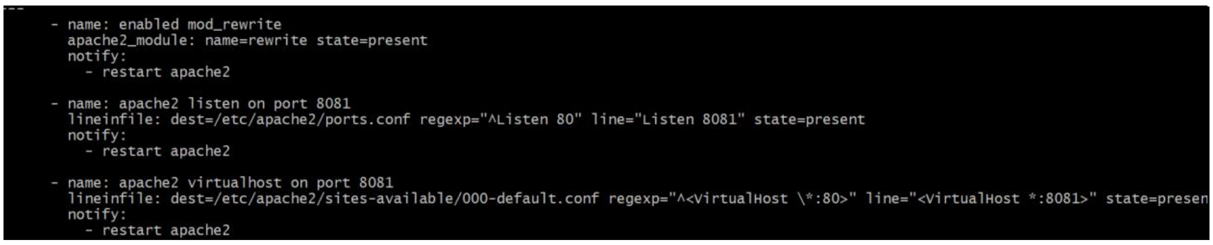
```
—
# tasks file for /etc/ansible/roles/apache
- import_tasks: install.yml
- import_tasks: configure.yml
- import_tasks: service.yml
```

```
$ cat tasks/install.yml
```



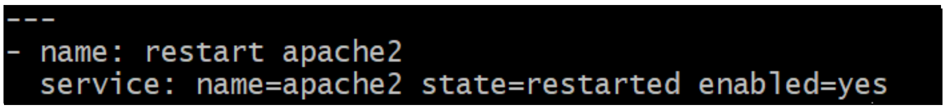
```
---
- name: install apache2
  apt: name=apache2 update_cache=yes state=latest
```

```
$ cat tasks/configure.yml
```



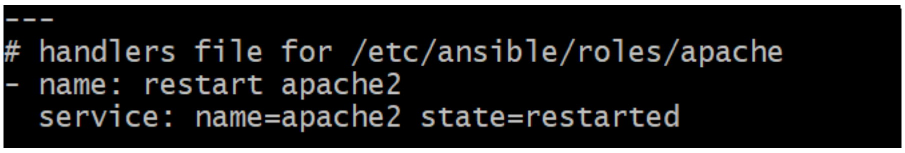
```
---
    - name: enabled mod_rewrite
      apache2_module: name=rewrite state=present
      notify:
        - restart apache2

    - name: apache2 listen on port 8081
      lineinfile: dest=/etc/apache2/ports.conf regexp="^Listen 80" line="Listen 8081" state=present
      notify:
        - restart apache2

    - name: apache2 virtualhost on port 8081
      lineinfile: dest=/etc/apache2/sites-available/000-default.conf regexp="^<VirtualHost \*:80>" line="<VirtualHost *:8081>" state=presen
      notify:
        - restart apache2
```

```
$ cat tasks/service.yml
```



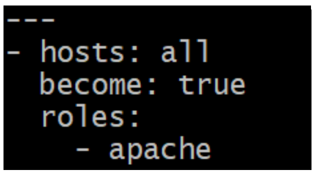
```
---
- name: restart apache2
  service: name=apache2 state=restarted enabled=yes
```

```
$cat handlers/main.yml
```



```
---
# handlers file for /etc/ansible/roles/apache
- name: restart apache2
  service: name=apache2 state=restarted
```

```
$cd /etc/ansible/
$ cat runsetup.yml
```



```
---
- hosts: all
  become: true
  roles:
    - apache
```

```
$ ansible-playbook  /etc/ansible/runsetup.yml
```

Dynamic inventory

1. Copy ec2.py and ec2.ini files
   https://github.com/vshn/ansible-dynamic-inventory-ec2/blob/master/ec2.py
   https://github.com/vshn/ansible-dynamic-inventory-ec2/blob/master/ec2.ini
2. Provide the execute permissions to above files
3. Attach a IAM role with ec2 access to ansible engine
4. Install python if required sudo apt-get install python-is-python3
5. Install pip

```
sudo apt-get -y install python3-pip
```

6. pip install boto

7. ansible-playbook ping.yaml -i ec2.py

Refer: https://clarusway.com/ansible-working-with-dynamic-inventory-using-aws-ec2-plugin/