ANSIBLE

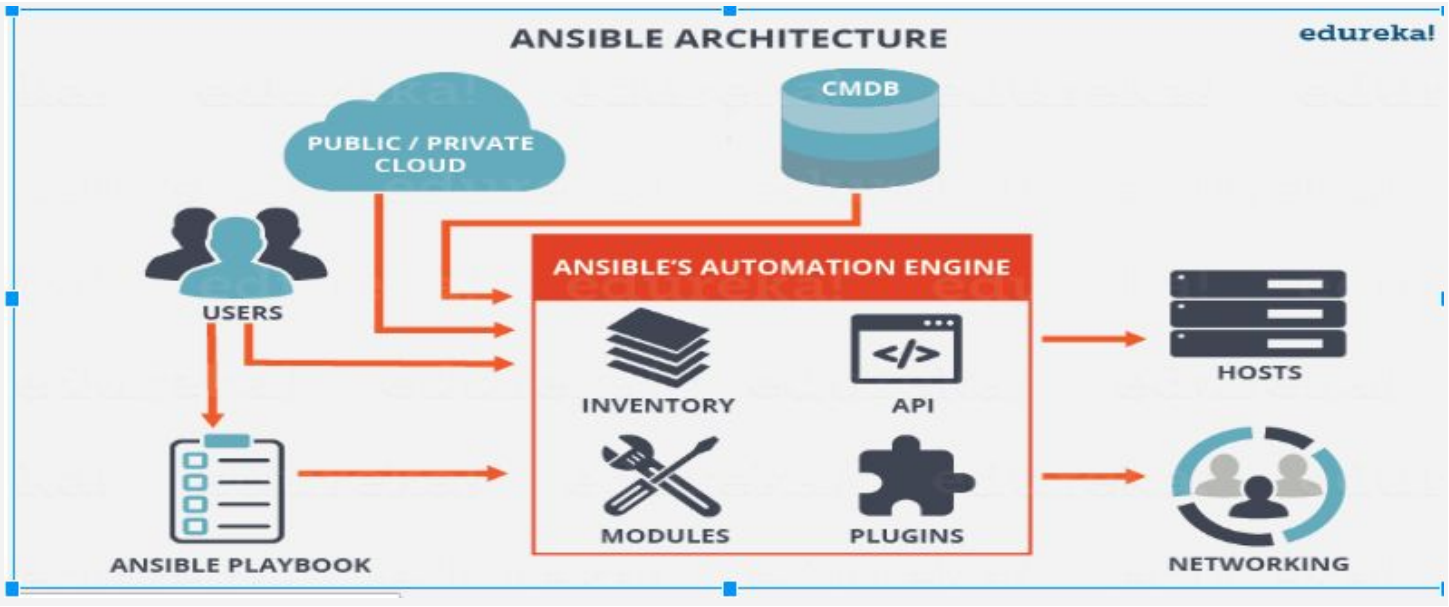
ANSIBLE(From Redhat):

1. Ansible is a radically simple IT automation engine that **automates cloud provisioning, configuration management, application deployment, intra-service orchestration**, and many other IT needs.
2. Ansible is an open source IT **Configuration Management, Deployment** & **Orchestration** tool. It aims to provide large productivity gains to a wide variety of automation challenges. This tool is very simple to use yet powerful enough to automate complex multi-tier IT application environments.
3. It uses **no agents** and no additional custom security infrastructure, so it's easy to deploy - and most importantly, it uses a very simple language (**YAML**, in the form of Ansible Playbooks) that allow you to describe your automation jobs in a way that approaches plain English.
4. Its architecture is combination of Modules, Plugins, Inventory and Playbooks.
5. **Ansible** is **PULL** model.(**Chef** and **Puppet** are **PUSH** model).
6. Ansible uses **Python** to process ansible playbooks written in yaml.(In chef and puppet **RUBY** is used).
7. Chef and Puppet uses Master and Agent model, we have to install agents(clients) on managed nodes of chef and puppet.
8. Ansible is **agent less**.
9. Ansible is **Idempotent(** If we run again and again ansible playbooks it wii not any effect to the environment**)**.
10. When we run ansible commands first it checks ansible configuration file.
11. Configuration file location is `/etc/ansible/ansible.cfg`
12. Involuntary file location is `/etc/ansible/hosts`
13. Primarily Ansible is Configuration Management Tool.
14. In ansible managed node can be web server, db server,load balancer or anything.
15. **Ansible Control Machine** is where we can trigger/run Ansible playbooks
16. **Ansible Target Machine/Managed Node** is the machine configured by Ansible. Example Web Server, DB server or Load Balancer.
17. Present Ansible **version 2.4(2.x).** Mine **2.0.0**
18. Ad Hoc commands are powerful, managing complex configurations using ad hoc is a difficult job rather u should use playbooks
19. Playbook contain set of tasks, tasks also called as plays.
20. Ansible playbooks are written in **Yaml.**
21. **Xml** and **json** are to use for exchange the data and configuration, but **yaml** is is used only for configuration.
22. Ansible internally uses **python**.
23. Ansible default do 5 tasks at a time.(config file) → forks

Other Configuration Management Tools:

1. Cheff
2. Puppet
3. Saltstack
4. etc..

Ansible Architecture:



Why Ansible?

1. Ansible is **agentless**(Main Difference between Chef and Puppet).
2. Ansible is Simple and Powerful tool.
3. Easy to learn and easy to use.
4. We can automate any task and manage thousand of servers.
5. Ansible is **open source**.
6. Ansible architecture is very simple
7. Highly durable and scalable.

Setting up Ansible Environment:

1. Ansible Control machine(Ubuntu)
2. Ansible Target Machine(Amazon Linux)

Ansible Control machine Requirements:

1. Currently Ansible can be run from any machine with **Python 2 (versions 2.6 or 2.7)** or **Python 3 (versions 3.5 and higher)** installed (**Windows isn't supported for the control machine**).

Ansible Target machine Requirements:

1. On the managed nodes, you need a way to communicate, which is normally **ssh**. By default this uses **sftp**. If that's not available, you can switch to **scp** in **ansible.cfg**. You also need **Python 2.6** or later.
2. **WinRM** is a management **protocol** used by **Windows** to remotely communicate with another server. It is a **SOAP-based protocol** that communicates over **HTTP/HTTPS**, and is included in all recent Windows operating systems.
3. Ansible uses the **pywinrm package** to communicate with Windows servers over **WinRM**. It is not installed by default with the Ansible package.
   But can be installed by running the following command

   pip install "pywinrm>=0.3.0"

Ansible Installation in Ubuntu:

1. sudo apt-get install software-properties-common
2. sudo apt-add-repository ppa:ansible/ansible
3. sudo apt-get update
4. sudo apt-get install ansible -y

Ansible Inventory: It contains managed node connection details plus some other variables. For example it contains IP address, username,password, private key file, Connection type, groups, variables etc..

Example:

```
52.40.19.210 ansible_user=ec2-user ansible_private_key_file=./ec2key.pem
```

1. Alias name:
   ```
   Web01 ansible_host=52.40.19.210 ansible_user=ec2-user ansible_private_key_file=./ec2key.pem
   ```
   Syntax: ansible web01 -m ping
2. Groups:
   ```
   [webservers]
   52.40.19.210 ansible_user=ec2-user ansible_private_key_file=./ec2key.pem
   152.40.19.210 ansible_user=ec2-user ansible_private_key_file=./ec2key.pem
   100.40.19.210 ansible_user=ec2-user ansible_private_key_file=./ec2key.pem
   ```
   Syntax: ansible webservers -m ping

3. Group variables:
   ```
   [webservers]
   52.40.19.210
   52.40.19.215
   52.40.19.211

   [webservers:vars]
    ansible_user=ec2-user
    ansible_private_key_file=./ec2key.pem
   ```

4. Group of groups:
   ```
   [group-1]
   52.40.19.2100
   [group-2]
   55.40.19.210

   [group:children]
   Group-1
   group-2
   ```

ansible.cfg: This is one more configuration file which contains details like location of inventory file, host_key_checking, library, ask_sudo_passwd, ask_calut_pass etc..

Note: When we run ansible commands first it looks for **ansible.cfg** and remaining details are gathered from this file. In forks(Threads) by default 5.

Example:

```
[defaults]
inventory=./inventory
```

Ansible connection types:

1. ssh → for linux servers
2. winrm → for windows
3. local → it is the connection type, playbook plays in same control machine itself

Ansible Facts: Ansible gathers some information from remote machines, that is called facts.

Syntax:

ansible 52.40.19.210 -m setup

Ansible conditions:

Disabling yes no while connecting to server:

1. Sudo vi `/etc/ansible/ansible.cfg`

```
# additional paths to search for roles in, colon separated
#roles_path    = /etc/ansible/roles

# uncomment this to disable SSH key host checking
host_key_checking = False

# change the default callback
#stdout_callback = skippy
# enable additional callbacks
```

How ansible locates ansible.cfg file:

Changes can be made and used in a configuration file which will be searched for in the following order:

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

Some Ansible Examples:

1. **ansible 52.40.19.210 -m ping**
   Where m=module, ping=module name
2. **ansible all -m command -a "touch /home/ec2-user/hello.sh" --become**
   Where m=module, command=module name, a=adhoc, become=as root user
3. **ansible all -m command -a "touch hello.sh" --become**
4. **ansible all -m yum -a 'name=httpd state=present' --become**
5. **ansible all -m command -a "service httpd start" --become**
6. **ansible all -m command -a "touch /var/www/html/index.html" --become**
7. **ansible all -m command -a "chkconfig httpd on" --become**
8. **ansible all -m shell -a "echo 'welcome'> /var/www/html/index.html" --become**
9. **ansible all -m command -a "cat /var/www/html/index.html" --become**
10. **ansible all -m service -a 'name=httpd state=started' --become**
11. **ansible all -m service -a 'name=httpd state=stopped' --become**
    [absent,present,installed,removed,restarted,reloaded]--> u will get under **service** module in document

12. **ansible 52.40.19.210 -m ping -i ./inventory** → passing inventory on run time
13. **Ansible-playbook users.yml --limit webservers[0]** → limited to web servers and 0th index
14. **Ansible-playbook users.yml --limit webservers[0:2]**
15. **ansible 52.40.19.210 -m setup** → This module is automatically called by playbooks to gather useful variables about remote hosts that can be used in playbooks
16. **ansible 34.214.170.242 -m setup -a "filter=ansible_os_family"** → to check os family of target machine
17. **Ansible all -m ping --forks=1**

Ansible-Playbooks:

Example-1:

```
---
- hosts: 34.214.170.242
  become: True
  tasks:
    - name: install apache
      yum:
        name: httpd
        state: present
    - name: start apache server
      service:
        name: httpd
        state: started
```

Example-2(With looping)

```
---
- hosts: 34.214.170.242
  become: True
  tasks:
    - name: install apache,docker,git
      yum:
        name: "{{item}}"
        state: present
      with_items:
        - git
        - docker
        - httpd
```

Example-3(Users with multiple loops):

```
---
- hosts: 34.214.170.242
  become: True
  tasks:
    - name: add several users
      user:
        name: "{{ item.name }}"
        state: present
```

```
      groups: "{{ item.groups }}"
    with_items:
      - { name: 'testuser1', groups: 'ec2-user' }
      - { name: 'testuser2', groups: 'root' }
```

Ansible dry run:

- Before applying the playbook, it may tell what will the result after applying the playbook like **terraform plan,** just for simulation

Syntax:

```
ansible-playbook users.yml --check
```

Ansible syntax-check: it will any syntactic issues

Syntax:

```
ansible-playbook --syntax-check users.yml
```

Ansible tags: To minimise execution time and can run particular task in the same playbook.

```
---
- hosts: all
  tasks:
    - name: task-1
      debug:
        msg: "performed task-1"
      tags:
        - task-1
    - name: task-2
      debug:
        msg: "performing task-2"
      tags:
        - task-2
```

1. **ansible -playbook tasks.yml**
2. **ansible -playbook tasks.yml --tags task-1**
3. **ansible -playbook tasks.yml --tags task-1 task-2**
4. **ansible -playbook tasks.yml --skip-tags task-2** → skip particular tag

Ansible debug module:

- Debug is the module which prints the messages on the console, like sysout.

Ansible limit: It will limit to the particular Servers or ip addresses. In real time when we apply to all web servers it will be fail on one or more servers, after that we want to re run this play book on particular field ips it will be use, because if u apply in all web servers it will take lot of time to execute, so we can save time.

Syntax:

1. **ansible -playbook tasks.yml --limit 172.136.85.69**
2. **ansible -playbook tasks.yml --limit 172.136.85.69,156.36.89.45**
3. **ansible -playbook tasks.yml **

> **--limit 172.136.85.69,156.36.89.45**

Ansible Includes:

Syntax:

```
---
- hosts: all
  become: True
  tasks:
    - include: amazon.yml
    - include: ubuntu.yml
```

Ansible Handlers: Handles are also ansible plays, they are executed when someone notifies.

Example:

```
---
- hosts: amazon
  become: True
  tasks:
    - include: amazon.yml
    - name: Start the  linux server
      service:
       name=httpd
       state=started
    - name: Enable apache on reboot of linux
      service:
       name=httpd
       enabled=yes
    - name: coping index.html on linux
      copy:
       src: index.html
       dest: /var/www/html
      notify:
        - restart apache server
  handlers:
    - name: restart apache server
      service:
       name: httpd
       state: restarted
```

Ansible Raw module:

1. The use case is installing python-simplejson on older (Python 2.6 and before) hosts that need it as a dependency to run modules, since nearly all core modules require it.

Example:

```
---
- hosts: 142.131.228.136
  become: true
  gather_facts: no
  tasks:
    - name: Installing Python on Ubuntu
      raw: apt install -y python-minimal
```