Ansible

Ansible is configuration management orchestration tool.It can configure systems, deploy software, and orchestrate advanced workflows to support application deployment, system updates, and more.
Ansible is based on the command line . Ansible tower or ansible awx is the Gui version of ansible.
But as of now only 90% work can be done using ansible tower.

Red hat acquired the ansible tool in 2015
Ansible is preferred mostly on linux systems but can work on windows also.

Why Ansible ?????

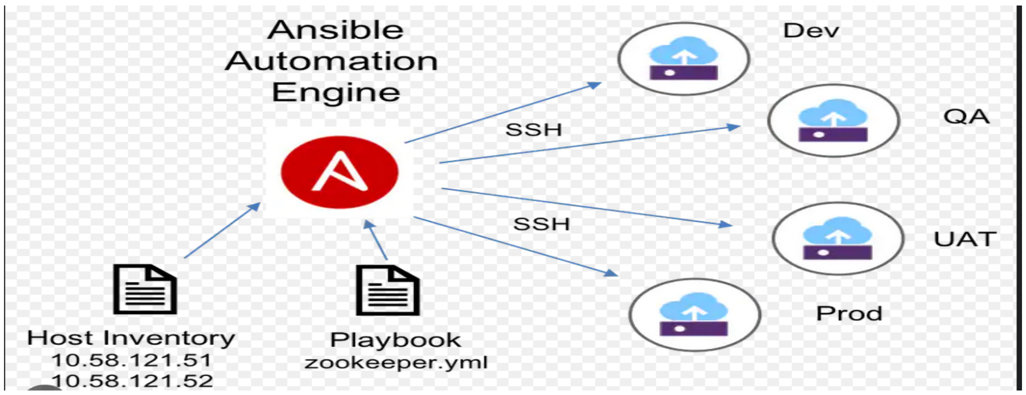
Let's take one scenario . In the company , there are multiple servers/systems that are located at different places.
Earlier system administrators manually accessed each and every system and configured, Install and update softwares and manage networks. That was a very tedious process.
Ansible just makes this process automated by writing a script and executing it so that it parallel installs or updates softwares in all the machines at same times.

Imagine how you as an administrator can update and install software in a company having 1000k Machines without ansible. You have to individually access each system which will take so many days for just installing a single software.

Other alternatives of Ansible in market are:
Chef , puppet

Some important points:

- Chef based on pull mechanism
- Ansible is based on a **push mechanism .** It means whenever any update requires we as server will push it in nodes/ clients . they do not have to come to the server each time to check if there is a new update required unlike chef.
  E.g - when you get a company laptop many times you get notifications of updating software. You don't ask the company for an update. They send notification of updates to every machine whenever they think it requires a new update.
- Ansible is agentless . here we do not have to install any software/agent in target nodes for communication with ansible servers unlike in chef. We only need ansible and python3 in the server machine and no specific software is required in clients/target machines.
- It directly communicates through ssh.
- It is very secure due to its agentless capabilities and open ssh security feature capability.



Inventory file - File that container details about your ansible clients/target machines that you as a server wants to communicate with.
You can make a group of clients for developers , or testers to manage in a better way.
By default it create inventory file in name of hosts file

Playbook - file which contains instructions or tasks to be executed in your client machine.
It is written in YAML format.

Ansible.cfg- It is an ansible configuration file where a certain setting of ansible is adjustable.
In this file you can also change the default path of the inventory file, playbook and other functionality depending on your requirement.

Module- Module is a command or set of similar commands to be executed on the client nodes.

Ansible server- The machine where ansible is installed from which all tasks and playbooks will be executed.

Host - Nodes which are automated by ansible .

Fact- Information fetched from the client system from the global variables with the gather fact operation.

Lab setup :

Here we are using our localhost/ local machine as an ansible server and we create and use containers which will be our nodes/ hosts/client machines.

Here we are using an ubuntu machine with python installed as server machine

For creating clients machines ,we are using an ubuntu container.
Use the below command to create and configure the client machine.

```
 docker run -it -d --name ubuntu ubuntu
 docker ps
 docker exec -it c732bb155a12 /bin/bash        ## add ur container_id in place of c732bb
 apt-get update
```

If ssh is not there,install and start using below command

```
apt install openssh-server
service ssh start
```

You need to install python3 in your client machine also if it is not there.

```
apt install python3
```

We are using ssh for communication because it's a safe method to access target machines instead of using passwords.

Ansible installation on server:

```
sudo apt update
sudo apt install ansible
```

Or

```
sudo apt remove ansible
sudo apt --purge autoremove
sudo apt update
sudo apt upgrade
sudo apt -y install software-properties-common
sudo apt-add-repository ppa:ansible/ansible
sudo apt install ansible
```

## to check ansible version

```
ansible --version
```

You can find the ansible.cfg configuration file inside /etc/ansible folder.
Go to that file using command `cd /etc/ansible/`
Here you find ansible.cfg file and host file/ inventory file. If host file is not there just create one host file using command `touch hosts`
Here the host file is an inventory file.

Default location of inventory file is /etc/ansible/hosts
You can also create your own inventory file.
Here in path /etc/ansible/ we can keep all our playbooks.

Configure Ansible to communicate with a remote node

We first create a user in the server and all client nodes . Because of security we should not give all credentials of root. Instead we create another user and give its user access and right to access only some function which is required for them and not complete access.
E.g when you join a company they don't give you complete access to company data. They just create your account and give that much access privilege that is required for you.

To create user use below command in both server and clients:

```
adduser ansible_usr            #ansible_usr is name of user you want to create
```

To give sudo privilege :

```
 usermod -aG sudo ansible_usr
```

If sudo is not working inside container, install sudo using below command:

```
apt install sudo
```

Now in server node, switch to ansible_usr using below command:

```
su ansible_usr
```

To generate ssh key in server

```
ssh-keygen
```

Now to save that public key to all client nodes use below commands in server node.

```
ssh-copy-id  ansible_usr@172.17.0..2
```

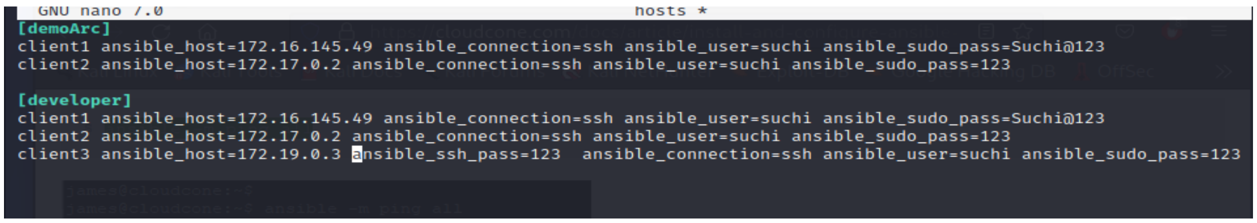
You have to give the ip address of your client node.

You can also cross-check if ssh working or not by using below command in server

```
ssh ansible_usr@172.17.0.2     ###give your client ip i.e here container ip not this ip
```

You can check ip address of container using command : `cat /etc/hosts`

Host file/ inventory file



```
  GNU nano 7.0                                    hosts *
[demoArc]
client1 ansible_host=172.16.145.49 ansible_connection=ssh ansible_user=suchi ansible_sudo_pass=Suchi@123
client2 ansible_host=172.17.0.2 ansible_connection=ssh ansible_user=suchi ansible_sudo_pass=123

[developer]
client1 ansible_host=172.16.145.49 ansible_connection=ssh ansible_user=suchi ansible_sudo_pass=Suchi@123
client2 ansible_host=172.17.0.2 ansible_connection=ssh ansible_user=suchi ansible_sudo_pass=123
client3 ansible_host=172.19.0.3 ansible_ssh_pass=123  ansible_connection=ssh ansible_user=suchi ansible_sudo_pass=123
```

Here, the first line is the group name. You can give any name .
Group name is not mandatory to give.
When we execute a playbook using a groupname . it will execute in all clients belonging to that group only.
You can create many groups as per your requirement.

You can also execute a playbook on a particular client node using its name like here client1 or client2.

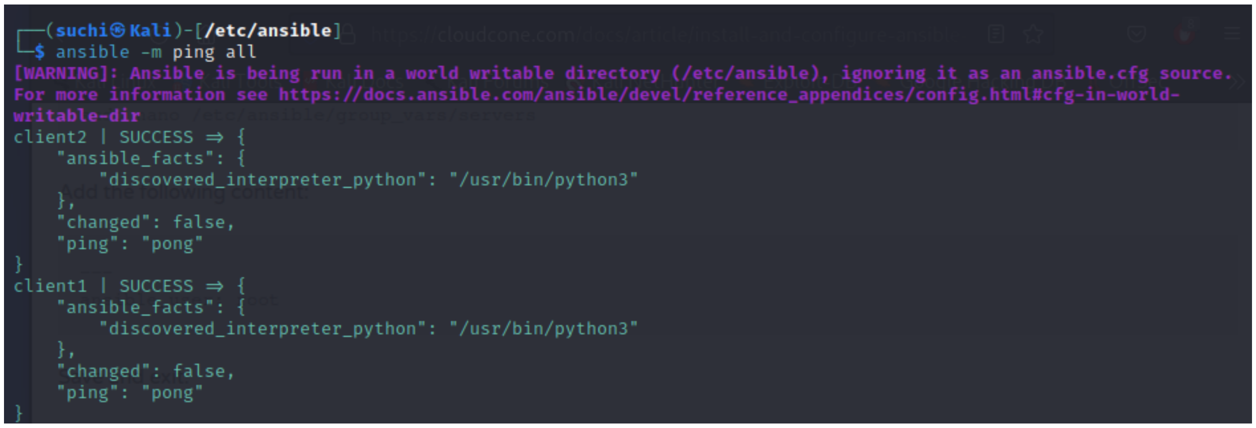
Testing connectivity to the servers

To test inventory file or check if it is able to communicate with clients nodes properly
Use below command:

ansible -m ping all
Here if you want to use your inventory file and not default file use below command
 ansible -i *myinventory* -m ping all   ##here myinventory is your inventory file name

-m means module , here we are using the ping module.
all means we want to execute a ping module in all client nodes present in the inventory file.
You can also pass **client node name** or **group name** in place of **all** to execute on particular client nodes or particular nodes inside that group.

Output of above command is shown below:



```
┌──(suchi㉿Kali)-[/etc/ansible]
└─$ ansible -m ping all
[WARNING]: Ansible is being run in a world writable directory (/etc/ansible), ignoring it as an ansible.cfg source.
For more information see https://docs.ansible.com/ansible/devel/reference_appendices/config.html#cfg-in-world-
writable-dir
client2 | SUCCESS => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python3"
    },
    "changed": false,
    "ping": "pong"
}
client1 | SUCCESS => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python3"
    },
    "changed": false,
    "ping": "pong"
}
```

You can see all the client nodes in inventory file using below command:

```
ansible all –list-hosts
```

Now , there are 3 ways to push or execute any work on client nodes:

- **Ad-hoc command**
  It can be run individually to perform quick functions.
  It is not used for configuration management and deployment as it can only automate single commands.
  We can't do everything using ad hoc commands.
  E.g `ansible all -a “ls”` here -a is argument

- **Ansible Modules**
  Predefined command stored in a module that can be executed directly on remote hosts.
  E.g `ansible all -m ping`

- **Playbooks**
  It is written in YAML language .
  Each playbook is composed of 1 or more modules.
  Here indentation is important.
  It is a blueprint of automation tasks which we execute with limited or no human involvement.
  Ansible Playbooks are executed on a set or group, or hosts, which together make up an Ansible inventory.
  They can describe a policy you want your remote systems to enforce, or a set of steps in a general IT process.
  It consists of vars,tasks , handlers , files, templates, roles but not everything is required to mention in the playbook each time.

Playbooks

Now , lets create 1 playbook ping.yml in same folder i.e /etc/ansible/ using below commands:

```
cd /etc/ansible/
touch ping.yml
nano ping.yml
```
here we are using nano editor , you can use any text editor

```
---
- name: this is just a test
  hosts: localhost
  gather_facts: true
  tasks:

  - name: ping test
    ping:
```

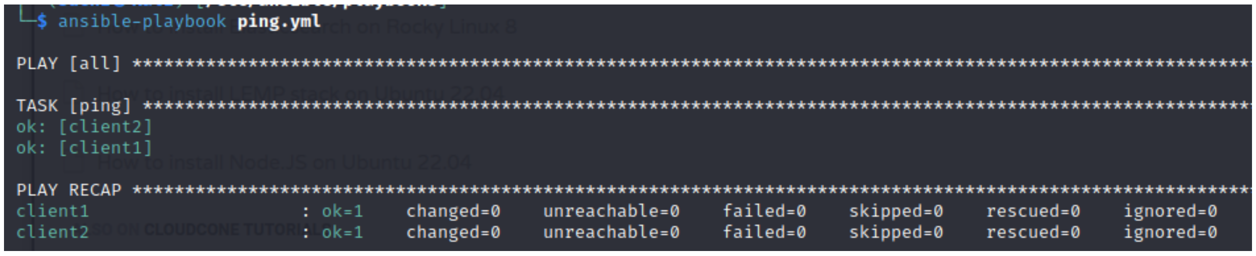
Playbook1

In **hosts** , you can pass either all or group name or any particular client node name to execute this playbook on that node only.
Here **name** is just like a message for users to debug and know which task of playbook is running and what this task is for.
In **tasks** , we mention the modules we want to execute on client nodes

To check syntax of playbook
ansible-playbook *demo2.yml* --syntax-check   ### demo2.yml is your playbook name

To execute playbook use below command:
ansible-playbook ping.yml   ## it by default consider all hosts in default inventory file
Or
ansible-playbook ping.yml --limit demoArc
Or
ansible-playbook -i *myinv* ping.yml   ## using -i you can give your own inventory file

Output:



```
└─$ ansible-playbook ping.yml

PLAY [all] ************************************************************************************

TASK [ping] ***********************************************************************************
ok: [client2]
ok: [client1]

PLAY RECAP ************************************************************************************
client1                    : ok=1    changed=0    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
client2                    : ok=1    changed=0    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
```

Conclusion

- This way you can create and try multiple playbooks.