Ansible playbook

* An **Ansible playbook** is a YAML file that contains a set of instructions (or "plays") for configuring, deploying, and managing servers or systems

* Ansible **ad-hoc commands** are a way to run simple, one-time tasks directly from the command line without the need for a playbook

* the **inventory file** specifies **where** (target machines), and the **playbook** specifies **what** (tasks) to be done on those machines.
* Inventory file we can create either in INI or YAML format

INI : Initialization File", no extension needed

YAML: required YAML extension

* Why are playbooks and inventory files made separately?  
  Your infrastructure can be dynamic (change IPs, add/remove servers) without affecting your automation tasks.

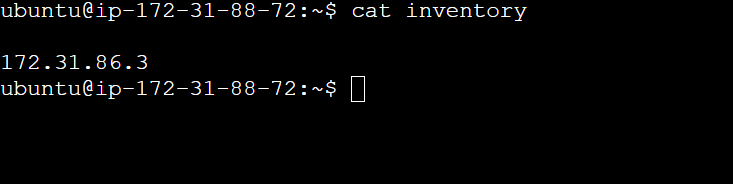
The playbook can be used across different environments or stages (dev, staging, production) simply by changing the inventory file.

ad-hoc Module

Modules are built-in functions in Ansible that perform specific tasks.

**Ansible ad-hoc commands**, and they use **Ansible modules** to perform tasks

1. **Shell Module**
2. Create an inventory file-> vim inventory
3. copy the target system ip either private or public



ubuntu@ip-172-31-88-72:~/.ssh$ ansible -i inventory 172.31.86.3 -m "shell"  -a "touch devopsclass"

172.31.86.3 | CHANGED | rc=0 >>

* -i inventory: This specifies the inventory file (inventory), which contains the list of hosts.
* -m specifies the module to use
* -a specifies the arguments to pass to the module
* CHANGED → Indicates that Ansible made a change (a new file was created).
* rc=0 → Return code 0 means the command executed successfully.

1. Since you did not specify a directory in the command (touch devopsclass), the file devopsclass will be created in the **home directory of the user** under which the Ansible command is being run on the target server (usually /home/ubuntu).
2. **With directory:**

ubuntu@ip-172-31-88-72:~/.ssh$ ansible -i inventory 172.31.86.3 -m "shell"  -a "touch /home/ubuntu/practise/devopsclass"

172.31.86.3 | CHANGED | rc=0 >>

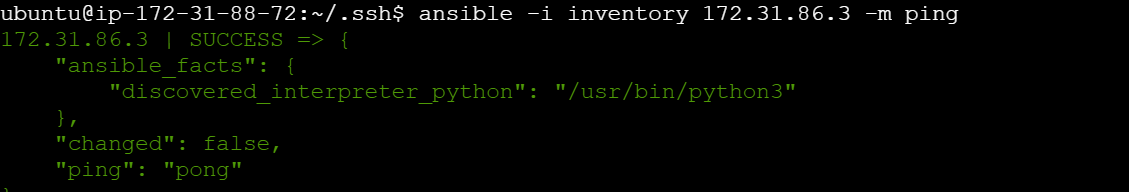
1. **To get number of core**

ubuntu@ip-172-31-88-72:~/.ssh$ ansible -i inventory 172.31.86.3 -m "shell"  -a nproc

           172.31.86.3 | CHANGED | rc=0 >>

1. **Ping Module**

* **Purpose**: To check the connectivity between the control node and the target node.



Or ansible all -m ping

1. **Copy Module**

copy module in Ansible is used to copy files (not directories)

ansible all -m copy -a "src=/path/to/local/file dest=/path/to/remote/file"

1. **Reboot Module**

   ansible all -m reboot

1. **Fetch Module**

ansible all -m fetch -a "src=/tmp/remote\_file dest=/tmp/local\_file flat=yes"

src=/tmp/remote\_file → The file on the target node that you want to fetch.

dest=/tmp/local\_file → Where to store the file on the control node.

flat=yes → Saves the file directly as local\_file or else mimic the remote directory structure

Fetches (downloads) a file from a remote machine to the control node (your local  system), Collecting logs, configuration files, or reports from multiple remote hosts

Copies (uploads) a file from the control node (your local system) to the remote machine, Deploying configuration files, scripts, or application files to remote hosts

1. **Shell Module with Multiple Commands**

ansible all -m shell -a "mkdir -p /tmp/newdir && touch /tmp/newdir/testfile"

When you use the -p option with the mkdir command, it **prevents errors** if the directory already exists.

1. **file Module**

file: Ensures that the path is a regular file (creates it if it doesn't exist).

directory: Ensures that the path is a directory (creates it if it doesn't exist).

absent: Ensures that the path is removed (either file or directory).

**state: file**: This will **ensure the file exists** and will also **check and set permissions**

**state: touch: This will ensure the file exists but will not check or modify the permissions,in the touch module, you cannot directly set the mode (permissions) of the file.**

* ansible all -m file -a "path=/tmp/mydirectory state=directory"

The state parameter is required because it explicitly defines the action that Ansible should take. Without it, Ansible wouldn't know whether to create, delete, or modify the file or directory.

* ansible all -m file -a "path=/tmp/myfile.txt state=touch"

It checks for the file /tmp/myfile.txt on all the target hosts defined in the inventory.

If the file does not exist, it will be created.

If the file already exists, it won’t change anything but will ensure it exists

* ansible all -m file -a "path=/tmp/myfile.txt state=absent"

This tells Ansible that the specified path should not exist. If the file or directory exists, Ansible will remove it

1. **Adhoc command to install any packages.**

ansible <target\_hosts> -m <package\_manager> -a "name=<package\_name> state=present" --become

ansible all -m apt -a "name=nginx state=present" --become

Nginx (**pronounced "Engine-X"**) is an **open-source web server**

1. **Adhoc command to uninstall any packages.**

ansible all -m apt -a "name=nginx state=absent" --become

**Ansible Ad-hoc Command to Start Nginx:**

ansible all -m service -a "name=nginx state=started enabled=true" --become

**enabled: true**: Nginx will start after a reboot.

**enabled: false**: Nginx will not start after a reboot.

**How to group servers in Ansible inventory:**

Before we saw that in the inventory file we copied ip of one system. Now suppose we have a number of systems.

You can group servers by defining them in the inventory file

A database server stores and manages data, while a web server delivers web content to users.

For example, in an **INI format**:**Initialization** file format,

ubuntu@ip-172-31-88-72:~$ cat inventory.ini

[web\_server]

172.31.86.3

[db\_server]

172.31.81.84

[all\_servers:children]

web\_server

db\_server

**ubuntu@ip-172-31-88-72:~$ ansible -i inventory.ini all\_servers -m ping**

172.31.86.3 | SUCCESS => {

    "ansible\_facts": {

        "discovered\_interpreter\_python": "/usr/bin/python3"

    },

    "changed": false,

    "ping": "pong"

}

172.31.81.84 | SUCCESS => {

    "ansible\_facts": {

        "discovered\_interpreter\_python": "/usr/bin/python3"

    },

    "changed": false,

    "ping": "pong"

}

ubuntu@ip-172-31-88-72:~$ ansible -i inventory.ini web\_server -m ping

172.31.86.3 | SUCCESS => {

    "ansible\_facts": {

        "discovered\_interpreter\_python": "/usr/bin/python3"

    },

    "changed": false,

    "ping": "pong"

}

For example, in an **YAML**  file format

ubuntu@ip-172-31-88-72:~$ cat inventory.yml

all:

  hosts:

    server1:

      ansible\_host: 172.31.86.3

      ansible\_user: ubuntu

    server2:

      ansible\_host: 172.31.81.84

      ansible\_user: ubuntu

  children:

    web\_server:

      hosts:

        server1:

        server2:

    db\_server:

      hosts:

        server2:

ubuntu@ip-172-31-88-72:~$ ansible -i inventory.yml web\_server -m ping

server1 | SUCCESS => {

    "ansible\_facts": {

        "discovered\_interpreter\_python": "/usr/bin/python3"

    },

    "changed": false,

    "ping": "pong"

}

server2 | SUCCESS => {

    "ansible\_facts": {

        "discovered\_interpreter\_python": "/usr/bin/python3"

    },

    "changed": false,

    "ping": "pong"

}

ubuntu@ip-172-31-88-72:~$ ansible -i inventory.yml all -m ping

server1 | SUCCESS => {

    "ansible\_facts": {

        "discovered\_interpreter\_python": "/usr/bin/python3"

    },

    "changed": false,

    "ping": "pong"

}

server2 | SUCCESS => {

    "ansible\_facts": {

        "discovered\_interpreter\_python": "/usr/bin/python3"

    },

    "changed": false,

    "ping": "pong"

}

Note:

If you're using an Ubuntu-based image, the default username is typically ubuntu.

## How to create PLAYBOOK? You can create a new YAML file (e.g., playbook.yml): vim playbook.yml

### Structure of a Playbook

.yml extension is imp

---

- name: Create directory and file using file and touch

  hosts: all

  become:true

  tasks:

    - name: Create directory using file module

      file:

        path: /tmp/mydir

        state: directory           # The file module creates the directory if the state is set to 'directory'

    - name: Create an empty file using the file module

      file:

        path: /tmp/mydir/myfile.txt

        state: touch          # Ensures the file is created (like the 'touch' command in Linux)

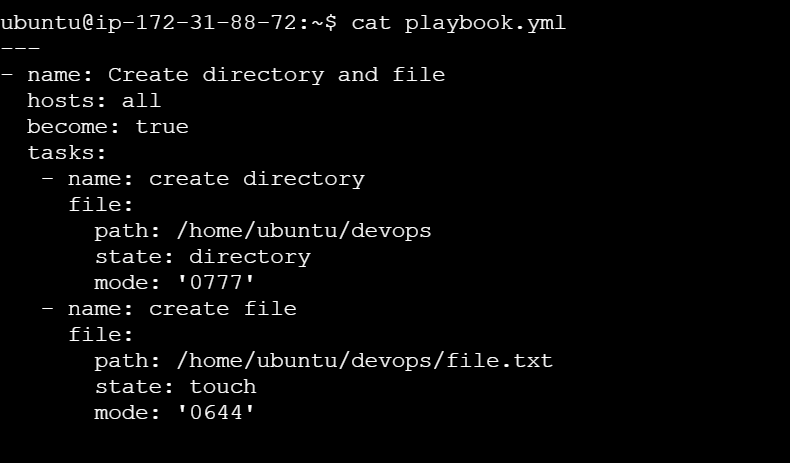
        mode: '0644'         # Sets permissions for the file (read/write for owner, read-only for others)

--- This indicates the beginning of the YAML document.

name: A name for your playbook, to describe the task you're performing.

hosts: Specifies which hosts or groups of hosts the playbook will run on. You can specify groups defined in the inventory (e.g., all, web\_servers, db\_servers).

true means that it will use sudo to execute the tasks as root. If the tasks don’t need elevated privileges, you can set this to false.



## How to run playbook? https://lh7-rt.googleusercontent.com/docsz/AD_4nXcgGhB2s_IgTinBK2CzcO4QJYnIvt--GMJvGN-PGiGxFMMcuP60MEwZLKJ2tCpusuudKEU8meoWcVrFjheyTlVIji7GcM-g0GYk_gYcavsduowxpU0q7FnU1wiPRqRWQINV5T2n?key=HecxA9SgWMbG4bzCe4AvyRBk

ansible-playbook -vvv -i inventory playbook.yml

When running with -vvv, the output will be much more detailed than a typical run:

* Information about each task being executed.
* Details on what files are being modified or installed.
* Any errors or issues that arise during playbook execution.
* The exact state of the hosts after each task

-v: Verbose - Provides more detailed output than the default, showing task results, the actions being performed, and any immediate output from the executed modules.

-vv: More verbose - In addition to the basic details, it shows debug-level information about what Ansible is doing internally, like what it is sending to the hosts and responses from them.

-vvv: Very verbose - This is the highest verbosity level. It shows all the output of a playbook, including