```
# ansible <group> -m <module> -a <arguments>
```

Here, we can also use a single host or all in place of <group> & <arguments> are optional to provide. Now let’s look at some basic commands to use with ansible,

Check connectivity of hosts

We have used this command in our previous tutorial also. The command to check connectivity of hosts is

```
# ansible <group> -m ping
```

Rebooting hosts

```
# ansible <group> -a “/sbin/reboot”
```

Checking host’s system information

Ansible collects the system’s information for all the hosts connected to it. To display the information of hosts, run

```
# ansible <group> -m setup | less
```

Secondly, to check a particular info from the collected information by passing an argument,

```
# ansible <group> -m setup -a “filter=ansible_distribution”
```

Transferring files

For transferring files, we use a module ‘copy’ & complete command that is used is

```
# ansible <group> -m copy -a “src=/home/ansible dest=/tmp/home”
```

Managing users

So to manage the users on the connected hosts, we use a module named 'user' & commands to use it are as follows,

Creating a new user

```
# ansible <group> -m user -a "name=ansible password=<encrypted
password>"
```

Deleting a user

```
# ansible <group> -m user -a "name=ansible state=absent"
```

Note: – To create an encrypted password, use the 'mkpasswd –method=sha-512' command.

Changing permissions & ownership

So for changing ownership of files of connected hosts, we use module named 'file' & commands used are

Changing permission of a file

```
# ansible <group> -m file -a "dest=/home/ansible/file1.txt
mode=777"
```

Changing ownership of a file

```
# ansible <group> -m file -a "dest=/home/ansible/file1.txt mode=777
owner=ansible group=ansible"
```

Managing Packages

So, we can manage the packages installed on all the hosts connected to ansible by using 'yum' & 'apt' modules & the complete commands used are

Check if package is installed & update it

```
# ansible <group> -m yum -a “name=httpd state=latest”
```

Check if package is installed & don’t update it

```
# ansible <group> -m yum -a “name= httpd state=present”
```

Check if package is at a specific version

```
# ansible <group> -m yum -a “name= httpd-1.8 state=present”
```

Check if package is not installed

```
# ansible <group> -m yum -a “name= httpd state=absent”
```

Managing services

So to manage services with ansible, we use modules ‘service’ & complete commands that are used are,

Starting a service

```
# ansible <group> -m service -a “name=httpd state=started”
```

Stopping a service

```
# ansible <group> -m service -a “name=httpd state=stopped”
```

Restarting a service

```
# ansible <group> -m service -a "name=httpd state=restarted"
```

Dry Run

Ansible supports running a playbook in dry run mode (also called Check Mode), in this mode, Ansible will **not** make any changes to your host, but simply report what changes would have been made if the playbook was run without this flag.

```
# ansible-playbook --check playbook.yml
```

Ansible Vault

If one of your tasks requires sensitive information (let's say the database user and password), it's a good practice to keep this information encrypted, instead of storing it in plain text.

Ansible ships with a command line tool called ansible-vault, that allows you to create and manage encrypted files. This way you can commit the encrypted file to your source control and only users with the decryption password will be able to read it.

```
# Encrypt an existing file. You'll need to create an encryption
password.

ansible-vault encrypt secrets.yml

# Creates a new, encrypted file. You'll need to create an
encryption password.

ansible-vault create secrets.yml
```

```
# Decrypt a file. You'll have to enter password used for
encryption.

# Use it with caution! Don't leave your files unecrypted.

ansible-vault decrypt secrets.yml

# Edit an encrypted file (uses vim by default, can be overriden by
the environment variable $EDITOR)

ansible-vault edit secrets.yml

# Print the contents of the encrypted file

ansible-vault edit secrets.yml
```

If you import the vars_file secrets.yml in your playbook, Ansible will fail, as it will not know how to read the encrypted file. You'll have to specify the command line argument –ask-vault-pass, which will make Ansible prompt you the password of the encrypted file.

```
# ansible-playbook playbook.yml -i hosts --ask-vault-password
```

Another way is to store the password in a file (which should not be committed) and specify the path to the file using the –vault-password-file argument. If this file is marked as executable, Ansible will run it and use the output as the password.

To make sure you have a root login to all listed hosts in inventory use below command;

```
# ansible all -m shell -a id
```

To run a Playbook using root,issue the below command:

```
ansible-playbook <playbookname> -k -K
```

tags

As your playbook gets bigger, it may get slower to run, and you may wish to run only part of a playbook to bypass the earlier steps.

Example:

```
tasks:

    - name: ensure package cache is up to date

      yum: update_cache=yes

      tags: install
```

```
# ansible-playbook webserver.yml -t install
```

Ansible When Statements:

Use the when condition to control whether a task or role runs or is skipped. This is normally used to change play behavior based on facts from the destination system.

Syntax

`when: (condition)`

Example

- `when: ansible_os_family == "Debian"`
- `when: ansible_pkg_mgr == "apt"`
- `when: myvariablename is defined`

Multiple Conditions

Syntax

- `When: condition1 and/or condition2`

Example (simple)

- `when: ansible_os_family == "Debian" and ansible_pkg_mgr == "apt"`

Magic Variable:

Using Magic Variable you can access information about your hosts.

Magic variable names are reserved - do not set variables with these names. The variable `environment` is also reserved.

The most commonly used magic variables are `hostvars`, `groups`, `group_names`, and `inventory_hostname`.

Consider this playbook:

```
- hosts: all
  tasks:
    - include: Ubuntu.yml
      when: ansible_os_family == "Ubuntu"

    - include: RHEL.yml
      when: ansible_os_family == "RedHat"
```

Example:

```
- hosts: all
  tasks:
    - name: Restart Apache on webservers
      become: yes
      service:
        name: apache2
        state: restarted
      when: webservers in group_names
```

Here group_names is a magic variable.

Importing Multiple Books:

```
[root@ansible-server ~]# cat global.yaml
```

```
---

# Combine multiple playbooks

  - import_playbook: play1.yaml

  - import_playbook: play2.yaml
```

Command Line Arguments:

We can pass values to playbook using arguments.

Example:
ansible-playbook release.yml --extra-vars "version=1.23.45 ther_variable=foo"