HowTo ASIX Ansible

Curs 2022-2023

Ansible Instal·lació i configuració d'un lab

- Ansible Documentation
  https://docs.ansible.com/ansible/latest/index.html

Descripció general

Eina per realitzar configuracions / desplegaments automatitzats a conjunts de hosts.

Ansible concepts:
https://docs.ansible.com/ansible/latest/getting_started/basic_concepts.html

Característiques principals:

- Agentless.
- Control node i managed nodes.
- Usa SSH per comunicar amb els hosts. Ha de disposar d'accés via SSH als hosts, per tant cal un usuari d'accés. Preferentment l'accés és via *ssh públic key*.
- Inventary: conjunts de hosts amb agrupacions.

Installation Ansible

- Ansible Installation Guide
  https://docs.ansible.com/ansible/latest/installation_guide/index.html

Installing on Debian 11 Bullseye

```
$ sudo vim /etc/apt/sources.list.d/ansible.list
deb http://ppa.launchpad.net/ansible/ansible/ubuntu focal main

$ sudo apt-key adv --keyserver keyserver.ubuntu.com --recv-keys
93C4A3FD7BB9C367

$ sudo apt-get update
$ sudo apt-get install ansible
```

```
$ ansible --version
ansible 2.9.21
  config file = /etc/ansible/ansible.cfg
  configured module search path = ['/home/ecanet/.ansible/plugins/modules',
'/usr/share/ansible/plugins/modules']
  ansible python module location = /usr/lib/python3.8/site-packages/ansible
```

```
  executable location = /usr/bin/ansible
  python version = 3.8.10 (default, May  4 2021, 00:00:00) [GCC 10.2.1 20201125
(Red Hat 10.2.1-9)]
```

Test d'ansible al propi host per verificar inventory

- El host local ha de tenir actiu el servei ssh.
- L'usuari actual ha de poder conectar al host local via ssh.

- Ansible Getting Started: https://docs.ansible.com/ansible/latest/getting_started/index.html

Exemple definite un inventory amb tres adreces IP del propi host correspondents al localhost, la ip pública i la de docker (si està instal·lat).

Exemple amb un inventory amb format YAML

```
$ cat localhost.yaml
myself:
  hosts:
    myloopback:
      ansible_host: 127.0.0.1
    mypublicip:
      ansible_host: 192.168.1.111
    mypublicdocker:
      ansible_host: 172.17.0.1

$ ansible all --list-hosts -i localhost.yaml
  hosts (3):
    myloopback
    mypublicip
    mypublicdocker
```

Exemple amb un inventory en format INI

```
$ cat localhost.ini
[myself]
127.0.0.1
192.168.1.111
172.17.0.1

$ ansible all --list-hosts -i localhost.ini
  hosts (3):
    127.0.0.1
    192.168.1.111
    172.17.0.1
```

Test al propi host per verificar ping dels managed nodes

Amb aquest exemple es verificarà que s'identifiquen correctament les adreces IP de l'inventory i que es disposa d'un usuari i accés SSH als managed nodes (als hosts on actuar).

Verificar que el servei ssh està activat. Per anar bé cal que el servei SSH ja disposi del fingerprint dels hosts destí en el known_hosts. Si no s'hi ha accedir mai podem fer-ho abans amb l'ordre:

```
$ sudo systemctl status sshd

$ ssh-keyscan 127.0.0.1 192.168.1.111 172.17.0.1
```

Realitzar un ping als hosts de l'inventari. S'utilitzen les opcions:

-m ping per indicar que ansible utilitzi el mòdul ping.

-k per indicar que ha de demanar el passwd de ssh interactivament (no és el que es pretendrà usualment, sinó fer-ho per pubkey).

-i inventary per indicar el fitxer amb l'inventary dels managed hosts

Observem que s'ha connectat correctament al host local per la interfície pública i pel loopback però no per la de docker.

```
$ ansible all -m ping -k -i localhost.yaml
SSH password: <password-de-l'usuari>

mypublicdocker | FAILED! => {
     "msg": "Using a SSH password instead of a key is not possible because Host Key checking is enabled and sshpass does not support this.  Please add this host's fingerprint to your known_hosts file to manage this host."
}

myloopback | SUCCESS => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python"
    },
    "changed": false,
    "ping": "pong"
}

mypublicip | SUCCESS => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python"
    },
    "changed": false,
    "ping": "pong"
}
```

Llistat de l'inventroy

```
$ ansible-inventory -i localhost.yaml --list
{
    "_meta": {
        "hostvars": {
            "myloopback": {
                "ansible_host": "127.0.0.1"
            },
            "mypublicdocker": {
                "ansible_host": "172.17.0.1"
            },
            "mypublicip": {
                "ansible_host": "192.168.1.111"
            }
        }
    },
    "all": {
        "children": [
            "myself",
            "ungrouped"
        ]
    },
    "myself": {
        "hosts": [
            "myloopback",
            "mypublicdocker",
            "mypublicip"
        ]
    }
}
```

```
$ ansible-inventory -i localhost.ini --list
{
    "_meta": {
        "hostvars": {}
    },
    "all": {
        "children": [
            "myself",
            "ungrouped"
        ]
    },
    "myself": {
        "hosts": [
            "127.0.0.1",
            "172.17.0.1",
            "192.168.1.111"
        ]
    }
}
```

Desplegar un lab per fer pràctiques

Procediment:

- Crear el directori de treball.
- Generar la parella de claus pública/privada de SSH.
- Generar el lab amb màquines virtuals desplegades amb Vagrant.
    - Crear un usuari ansible a cada màquina.
    - Posar la clau pública generada al pas-2 al *athorized_keys*.
- Verificar l'accés a les VM tant via *vagrant ssh* com fent ssh amb l'usuari ansible.

Generar una parella de claus SSH

Primerament es generarà una parella de claus SSH (una clau pública i una de privada) usant ssh-keygen. En la ruta indicar que els fitxers estiguin en el directori actiu i amb el nom ansible per identificar que són les que s0utilitzaran en aquest lab. Aquestes caus es creen sense passphrase (de moment…)

```
$ ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/home/ecanet/.ssh/id_rsa): ansible_key
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in ansible_key
Your public key has been saved in ansible_key.pub
The key fingerprint is:
SHA256:/AvumLX0RVNHrfRRf4BndIih7YpnnSFyykZTVR/Jw8s ecanet@mylaptop.edt.org

$ ls -l
-rw-------. 1 ecanet ecanet 2610 Feb 14 17:28 ansible_key
-rw-r--r--. 1 ecanet ecanet  577 Feb 14 17:28 ansible_key.pub
-rw-rw-r--. 1 ecanet ecanet   45 Feb 14 17:05 localhost.ini
-rw-rw-r--. 1 ecanet ecanet  164 Feb 14 17:03 localhost.yaml
-rw-rw-r--. 1 ecanet ecanet  844 Feb 14 17:26 Vagrantfile
```

Generar les màquines virtuals amb Vagrantfile

Usar un fitxer de desplegament Vagrantfile com el següent, que crea un servidor tres workers Debian 11.

```
# -*- mode: ruby -*-
# vi: set ft=ruby :

Vagrant.configure("2") do |config|
  config.vm.synced_folder ".", "/vagrant"

  config.vm.provider "virtualbox" do |vb|
    vb.memory = 4096
    vb.cpus = 2
  end

  config.vm.define "server" do |srv|
    srv.vm.box = "debian/bullseye64"
    srv.vm.network "public_network", bridge: "enp1s0"
    srv.vm.provision "shell", inline: $install_ssh_pubkey_script_debian

  end

  (1..3).each do |i|
    config.vm.define "worker#{i}" do |wk|
      #wk.vm.box = "generic/alpine38"
      wk.vm.box = "debian/bullseye64"
      wk.vm.network "public_network", bridge: "enp1s0"
      wk.vm.provision "shell", inline: $install_ssh_pubkey_script_debian
    end
  end
end

$install_ssh_pubkey_script_debian = <<SCRIPT
sudo useradd -m -s /bin/bash ansible
sudo mkdir /home/ansible/.ssh
sudo cp /vagrant/ansible_key.pub /home/ansible/.ssh/authorized_keys
sudo chown -R ansible.ansible /home/ansible/.ssh/
SCRIPT
```

Recordeu que vagrant ja crea un usuari dins de les VM i crea també una parella de claus pública/privada (insegura) per accedir via ssh. A més a més propaga el port 22 de la VM al port 2222 del host amfitrió.

```
==> server: Forwarding ports...
    server: 22 (guest) => 2222 (host) (adapter 1)
```

```
vagrant@bullseye:~$ ls -la .ssh/
-rw------- 1 vagrant vagrant  389 Feb 14 16:59 authorized_keys

vagrant@bullseye:~$ cat .ssh/authorized_keys
ssh-rsa
AAAAB3NzaC1yc2EAAAADAQABAAABAQCpJ3j8+eQ7FsZAe9z8d1v7Vi3npxtRg/szLwSKExPmKUXagqoTR
2a4e62rYadkdmHBYEofkA7PSgtmWbStK5zWH/jwYtZp3hrRgK+ePuRXPzLzmkCGhmlIP/YR+5CtmmSZcB
LOpOlGeeVBXRr5YpDJK6pMvGms93W9Jdacw/PYkPbt62V8rDb5obcbwgGGmBvCJiMX1Mlf8UMe+f1MFkI
bKvCyt5Un3Yo9dl6QQ0F+ddacT3aQzd4gEJRYUio7goKwJ2sOGRRcmPEiqXc5SdwLQMB3l/H5IpYajZxd
jFWXm2RsVOvShW/YIXHfiiNTw12edHH65QrbWeewJ1lktrod vagrant

vagrant@bullseye:~$ nmap localhost
Starting Nmap 7.80 ( https://nmap.org ) at 2023-02-14 17:04 UTC
Nmap scan report for localhost (127.0.0.1)
Host is up (0.000070s latency).
Other addresses for localhost (not scanned): ::1
Not shown: 999 closed ports
PORT   STATE SERVICE
22/tcp open  ssh
```

Crear l'usuari ansible i posar la clau pública al authorizd_keys

Per poder accedir amb ansible a cada VM es requereix un usuari que permeti l'accés a la màquina via SSH preferentment amb accés per clau pública. En aquest lab es crea a cada màquina un usuari ansible i es posa la clau pública (generada al pas 2) dins del fitxer *authorized_keys* de l'usuari ansible.
Observem la part de codi a modificar/afegir al vagrantfile:

```
config.vm.define "server" do |srv|
  …
  srv.vm.provision "file", source: "./ansible_key.pub", \
                   destination: "./.ssh/authorized_keys"
end

$install_ssh_pubkey_script_debian = <<SCRIPT
sudo useradd -m -s /bin/bash ansible
sudo mkdir /home/ansible/.ssh
sudo cp /vagrant/ansible_key.pub /home/ansible/.ssh/authorized_keys
sudo chown -R ansible.ansible /home/ansible/.ssh/
SCRIPT
```

Verificar l'accés amb l'usuari vagrant i amb l'usuari ansible

Podem verificar que un cop iniciada la màquina server s'hi pot accedir normalment amb vagrant:

```
$ vagrant up

$ vagrant ssh server
Linux bullseye 5.10.0-20-amd64 #1 SMP Debian 5.10.158-2 (2022-12-13) x86_64

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.

vagrant@bullseye:~$ ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default
qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP
group default qlen 1000
    link/ether 08:00:27:8d:c0:4d brd ff:ff:ff:ff:ff:ff
    altname enp0s3
    inet 10.0.2.15/24 brd 10.0.2.255 scope global dynamic eth0
       valid_lft 86353sec preferred_lft 86353sec
    inet6 fe80::a00:27ff:fe8d:c04d/64 scope link
       valid_lft forever preferred_lft forever
3: eth1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP
group default qlen 1000
    link/ether 08:00:27:ce:9a:05 brd ff:ff:ff:ff:ff:ff
    altname enp0s8
    inet 192.168.1.39/24 brd 192.168.1.255 scope global dynamic eth1
       valid_lft 43160sec preferred_lft 43160sec
    inet6 fe80::a00:27ff:fece:9a05/64 scope link
       valid_lft forever preferred_lft forever
```

I també que es pot accedir amb l'usuari ansible

```
$ ssh -i ansible_key -p 2222 ansible@localhost
The authenticity of host '[localhost]:2222 ([127.0.0.1]:2222)' can't be
established.
ECDSA key fingerprint is SHA256:OZBI7P++lnBkHD8lnxb6LTYq6uL35eOUOvs2/EEYPAc.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
```

```
Warning: Permanently added '[localhost]:2222' (ECDSA) to the list of known hosts.
Linux bullseye 5.10.0-20-amd64 #1 SMP Debian 5.10.158-2 (2022-12-13) x86_64

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.

ansible@bullseye:~$ id
uid=1001(ansible) gid=1001(ansible) groups=1001(ansible)
```

```
$ ssh -i ansible_key -p 22 ansible@192.168.1.39
The authenticity of host '192.168.1.39 (192.168.1.39)' can't be established.
ECDSA key fingerprint is SHA256:OZBI7P++lnBkHD8lnxb6LTYq6uL35eOUOvs2/EEYPAc.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '192.168.1.39' (ECDSA) to the list of known hosts.
Linux bullseye 5.10.0-20-amd64 #1 SMP Debian 5.10.158-2 (2022-12-13) x86_64

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Tue Feb 14 17:29:39 2023 from 10.0.2.2
ansible@bullseye:~$
```

```
$ vagrant  status
Current machine states:
server                    running (virtualbox)
worker1                   running (virtualbox)
worker2                   running (virtualbox)
worker3                   running (virtualbox)
This environment represents multiple VMs. The VMs are all listed
above with their current state. For more information about a specific
VM, run `vagrant status NAME`.
```

Probar ansible al lab de vagrant

Primerament caldrà desplegar el lab. Atenció: el lab desplegat usa adreces públiques del bridge, per tant primer caldrà identificar les adreces IP dels hosts virtuals.

Procediment:

- Desplegar el lab amb vagrant.
- Identificar les adreces IP de les màquines virtuals.
- Generar l'inventary amb aquestes adreces.
- Aplicar amb vagrant un mòdul, per exemple el mòdul ping.

Desplegar el lab, iniciar sessions i anotar adreces ip

```
$ vagrant up

$ vagrant ssh server
vagrant@bullseye:~$ ip a

$ vagrant ssh worker1
```

```
$ vagrant ssh worker2
$ vagrant ssh worker3
```

Crear l'inventory:

```
$ cat inventory_lab_public.yaml
servernet:
  hosts:
    server:
      ansible_host: 192.168.1.55

workersnet:
  hosts:
    worker1:
      ansible_host: 192.168.1.46
    worker2:
      ansible_host: 192.168.1.47
    worker3:
      ansible_host: 192.168.1.50

vagrantlabpublic:
  children:
    servernet:
    workersnet:
```

```
$ ansible-inventory -i inventory_lab_public.yaml --list
{
    "_meta": {
        "hostvars": {
            "server": {
                "ansible_host": "192.168.1.55"
            },
            "worker1": {
                "ansible_host": "192.168.1.46"
            },
            "worker2": {
                "ansible_host": "192.168.1.47"
            },
            "worker3": {
                "ansible_host": "192.168.1.50"
            }
        }
…
```

Problema: per accedir als managed hosts amb ansible hem creat dins d'ells un usuari ansible i disposem de la clau privada que en permet l'accés, però cal poder-li indicar l'usuari que es vol usar (si no utilitza l'usuari de la sessió actual) i on està el fitxer de clau privada.

Observem que amb l'usuari actaul no funciona i demana el password ssh:

```
$ ansible servernet -m ping -k -i inventory_lab_public.yaml
SSH password:  [ERROR]: User interrupted execution
```

Podem consultar la Ansible cheatsheet

- https://docs.ansible.com/ansible/latest/command_guide/cheatsheet.html
- https://docs.ansible.com/ansible/latest/cli/ansible.html

En l'exemple s'egüent s'utilitzen les següents opcions:

servernet (no és una opcio) argument que indica els managed hosts a usar.

-u ansible per indicar que l'usuari destí és l'usuari ansible
--private-key ./ansible_key clau provada a usar en la connexió ssh
-i inventory_lab_public.yaml inventory de hosts
-m ping mòdul directe a utilitzar

```
$ ansible servernet  -u ansible --private-key ./ansible_key \
                     -i inventory_lab_public.yaml -m ping
server | SUCCESS => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python3"
    },
    "changed": false,
    "ping": "pong"
}
```

En l'exemple anterior s'ha fer ping als managed hosts de la xarxa *servernet*. Anem a provar ara als de la xarxa *workersnet*.

```
$ ansible workersnet  -u ansible --private-key ./ansible_key \
                      -i inventory_lab_public.yaml -m ping
worker3 | SUCCESS => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python3"
    },
    "changed": false,
    "ping": "pong"
}
worker1 | SUCCESS => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python3"
    },
    "changed": false,
    "ping": "pong"
}
worker2 | SUCCESS => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python3"
    },
    "changed": false,
    "ping": "pong"
}
```

I si provem tots els hosts definitions a l'inventory obtenim:

```
$ ansible all  -u ansible --private-key ./ansible_key -i inventory_lab_public.yaml
-m ping
worker3 | SUCCESS => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python3"
    },
    "changed": false,
    "ping": "pong"
}
worker2 | SUCCESS => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python3"
    },
    "changed": false,
    "ping": "pong"
}
server | SUCCESS => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python3"
    },
    "changed": false,
    "ping": "pong"
}

worker1 | SUCCESS => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python3"
    },
    "changed": false,
    "ping": "pong"
}
```

Playbooks

- Getting Started: Creating a playbook
  https://docs.ansible.com/ansible/latest/getting_started/get_started_playbook.html

- Using Ansible playbooks
  https://docs.ansible.com/ansible/latest/playbook_guide/index.html

- CLI ansible-playbook cheatsheet
  https://docs.ansible.com/ansible/latest/cli/ansible-playbook.html

Conceptes:

```
Playbook
A list of plays that define the order in which Ansible performs operations, from
top to bottom, to achieve an overall goal.

Play
An ordered list of tasks that maps to managed nodes in an inventory.

Task
A list of one or more modules that defines the operations that Ansible performs.

Module
A unit of code or binary that Ansible runs on managed nodes. Ansible modules are
grouped in collections with a Fully Qualified Collection Name (FQCN) for each
module.
```

Exemples d'ordres:

- ansible-playbook -u ansible --private-key ./ansible_key -i inventory_lab_public.yaml playbook_exemple_01_ping.yaml
- ansible-playbook -C sampleplaybook.yml -i ansible_hosts
- ansible-playbook – syntax-check sampleplaybook.yml -i ansible_hosts

Exemple-1 Playbook: fer ping als managed hosts

En aquest exemple es genera un playbook molt senzill per fer ping als managed hosts.

```
$ cat playbook_exemple_01_ping.yaml
- name: My first play
  hosts: workersnet
  tasks:
   - name: Ping my hosts
     ansible.builtin.ping:
   - name: Print message
     ansible.builtin.debug:
       msg: Hello world
```

```
$ ansible-playbook -u ansible --private-key ./ansible_key \
                    -i inventory_lab_public.yaml playbook_exemple_01_ping.yaml

PLAY [My first play]
*****************************************************************************************************************
*************************

TASK [Gathering Facts]
*****************************************************************************************************************
**********************
ok: [worker3]
ok: [worker2]
ok: [worker1]

TASK [Ping my hosts]
*****************************************************************************************************************
************************
ok: [worker2]
ok: [worker1]
ok: [worker3]

TASK [Print message]
*****************************************************************************************************************
************************
ok: [worker1] => {
    "msg": "Hello world"
}
ok: [worker2] => {
    "msg": "Hello world"
}
ok: [worker3] => {
    "msg": "Hello world"
}

PLAY RECAP
*****************************************************************************************************************
**********************************
worker1                    : ok=3    changed=0    unreachable=0    failed=0    skipped=0    rescued=0
ignored=0
worker2                    : ok=3    changed=0    unreachable=0    failed=0    skipped=0    rescued=0
ignored=0
worker3                    : ok=3    changed=0    unreachable=0    failed=0    skipped=0    rescued=0
ignored=0
```

Exemple-2 Playbook: update web servers

```
---
- name: Update web servers
  hosts: webservers
  remote_user: root

  tasks:
  - name: Ensure apache is at the latest version
    ansible.builtin.yum:
      name: httpd
      state: latest
  - name: Write the apache config file
    ansible.builtin.template:
      src: /srv/httpd.j2
      dest: /etc/httpd.conf

- name: Update db servers
  hosts: databases
  remote_user: root

  tasks:
  - name: Ensure postgresql is at the latest version
    ansible.builtin.yum:
      name: postgresql
      state: latest
  - name: Ensure that postgresql is started
    ansible.builtin.service:
      name: postgresql
      state: started
```

Ansible builtin

- Ansible builtin
  https://docs.ansible.com/ansible/latest/collections/ansible/builtin/index.html

```
# Modules
```

- add_host module – Add a host (and alternatively a group) to the ansible-playbook in-memory inventory
- apt module – Manages apt-packages
- apt_key module – Add or remove an apt key
- apt_repository module – Add and remove APT repositories
- assemble module – Assemble configuration files from fragments
- assert module – Asserts given expressions are true
- async_status module – Obtain status of asynchronous task
- blockinfile module – Insert/update/remove a text block surrounded by marker lines
- command module – Execute commands on targets
- copy module – Copy files to remote locations
- cron module – Manage cron.d and crontab entries
- debconf module – Configure a .deb package
- debug module – Print statements during execution
- dnf module – Manages packages with the *dnf* package manager
- dpkg_selections module – Dpkg package selection selections
- expect module – Executes a command and responds to prompts
- fail module – Fail with custom message
- fetch module – Fetch files from remote nodes
- file module – Manage files and file properties
- find module – Return a list of files based on specific criteria
- gather_facts module – Gathers facts about remote hosts
- get_url module – Downloads files from HTTP, HTTPS, or FTP to node
- getent module – A wrapper to the unix getent utility
- git module – Deploy software (or files) from git checkouts
- group module – Add or remove groups
- group_by module – Create Ansible groups based on facts
- hostname module – Manage hostname
- import_playbook module – Import a playbook

- import_role module – Import a role into a play
- import_tasks module – Import a task list
- include module – Include a task list
- include_role module – Load and execute a role
- include_tasks module – Dynamically include a task list
- include_vars module – Load variables from files, dynamically within a task
- iptables module – Modify iptables rules
- known_hosts module – Add or remove a host from the `known_hosts` file
- lineinfile module – Manage lines in text files
- meta module – Execute Ansible 'actions'
- package module – Generic OS package manager
- package_facts module – Package information as facts
- pause module – Pause playbook execution
- ping module – Try to connect to host, verify a usable python and return `pong` on success
- pip module – Manages Python library dependencies
- raw module – Executes a low-down and dirty command
- reboot module – Reboot a machine
- replace module – Replace all instances of a particular string in a file using a back-referenced regular expression
- rpm_key module – Adds or removes a gpg key from the rpm db
- script module – Runs a local script on a remote node after transferring it
- service module – Manage services
- service_facts module – Return service state information as fact data
- set_fact module – Set host variable(s) and fact(s).
- set_stats module – Define and display stats for the current ansible run
- setup module – Gathers facts about remote hosts
- shell module – Execute shell commands on targets
- slurp module – Slurps a file from remote nodes
- stat module – Retrieve file or file system status
- subversion module – Deploys a subversion repository
- systemd module – Manage systemd units
- systemd_service module – Manage systemd units
- sysvinit module – Manage SysV services.
- tempfile module – Creates temporary files and directories
- template module – Template a file out to a target host
- unarchive module – Unpacks an archive after (optionally) copying it from the local machine
- uri module – Interacts with webservices
- user module – Manage user accounts
- validate_argument_spec module – Validate role argument specs.
- wait_for module – Waits for a condition before continuing
- wait_for_connection module – Waits until remote system is reachable/usable
- yum module – Manages packages with the *yum* package manager
- yum_repository module – Add or remove YUM repositories

Collections

- https://docs.ansible.com/ansible/latest/collections/index.html

```
#Collections
```

- amazon.aws
- ansible.builtin
- ansible.netcommon
- ansible.posix
- ansible.utils
- ansible.windows
- arista.eos
- awx.awx
- azure.azcollection
- check_point.mgmt
- chocolatey.chocolatey
- cisco.aci
- cisco.asa
- cisco.dnac
- cisco.intersight
- Cisco.ios
- …

Using variables

- Using variables
  https://docs.ansible.com/ansible/latest/playbook_guide/playbooks_variables.html

Exemples Ansible documentation

```
---
- name: Update web servers
  hosts: webservers
  remote_user: root

  tasks:
  - name: Ensure apache is at the latest version
    ansible.builtin.yum:
      name: httpd
      state: latest
  - name: Write the apache config file
    ansible.builtin.template:
      src: /srv/httpd.j2
      dest: /etc/httpd.conf

- name: Update db servers
  hosts: databases
  remote_user: root

  tasks:
  - name: Ensure postgresql is at the latest version
    ansible.builtin.yum:
```

```
      name: postgresql
      state: latest
  - name: Ensure that postgresql is started
    ansible.builtin.service:
      name: postgresql
      state: started
```

Exemples generals (altres exemples)

https://spacelift.io/blog/ansible-tutorial

```
---
- name: Intro to Ansible Playbooks
  hosts: all

  tasks:
  - name: Copy file hosts with permissions
    ansible.builtin.copy:
      src: ./hosts
      dest: /tmp/hosts_backup
      mode: '0644'
  - name: Add the user 'bob'
    ansible.builtin.user:
      name: bob
    become: yes
    become_method: sudo
  - name: Upgrade all apt packages
    apt:
      force_apt_get: yes
      upgrade: dist
    become: yes
```

https://spacelift.io/blog/ansible-tutorial

```
---
- name: Variables playbook
  hosts: all
  vars:
      state: latest
      user: bob
  tasks:
  - name: Add the user {{ user }}
    ansible.builtin.user:
      name: "{{ user }}"
  - name: Upgrade all apt packages
    apt:
      force_apt_get: yes
      upgrade: dist
  - name: Install the {{ state }} of package "nginx"
    apt:
      name: "nginx"
      state: "{{ state }}"
```

https://www.middlewareinventory.com/blog/ansible-playbook-example/

```
---
  - name: Playbook
    hosts: webservers
    become: yes
    become_user: root
    tasks:
      - name: ensure apache is at the latest version
        yum:
          name: httpd
          state: latest
      - name: ensure apache is running
        service:
```

```
          name: httpd
          state: started
```

https://www.middlewareinventory.com/blog/ansible-playbook-example/

```
---
  # Play1 - WebServer related tasks
  - name: Play Web - Create apache directories and username in web servers
    hosts: webservers
    become: yes
    become_user: root
    tasks:
      - name: create username apacheadm
        user:
          name: apacheadm
          group: users,admin
          shell: /bin/bash
          home: /home/weblogic

      - name: install httpd
        yum:
          name: httpd
          state: installed

  # Play2 - Application Server related tasks
  - name: Play app - Create tomcat directories and username in app servers
    hosts: appservers
    become: yes
    become_user: root
    tasks:
      - name: Create a username for tomcat
        user:
          name: tomcatadm
          group: users
          shell: /bin/bash
          home: /home/tomcat

      - name: create a directory for apache tomcat
        file:
          path: /opt/oracle
          owner: tomcatadm
          group: users
          state: present
          mode: 0755
```

https://www.middlewareinventory.com/blog/ansible-playbook-example/

Usar variables

```
---
  - name: Playbook
    hosts: webservers
    become: yes
    become_user: root
    vars:
       key_file:  /etc/apache2/ssl/mywebsite.key
       cert_file: /etc/apache2/ssl/mywebsite.cert
       server_name: www.mywebsite.com
    tasks:
      - name: ensure apache is at the latest version
        yum:
          name: httpd
          state: latest
      ### SOME MORE TASKS WOULD COME HERE ###
      # you can refer the variable you have defined earlier like this #
      # "{{key_file}}"  (or) "{{cert_file}}" (or) "{{server_name}}" #
      ##
      - name: ensure apache is running
        service:
          name: httpd
          state: started
```

https://www.middlewareinventory.com/blog/ansible-playbook-example/

Usar variables en fitxers

```
---
  - name: Playbook
    hosts: webservers
    become: yes
    become_user: root
    vars_files:
        - apacheconf.yml
    tasks:
      - name: ensure apache is at the latest version
        yum:
          name: httpd
          state: latest
      ### SOME MORE TASKS WOULD COME HERE ###
      # you can refer the variable you have defined earlier like this #
      # "{{key_file}}"  (or) "{{cert_file}}" (or) "{{server_name}}" #
      ##
      - name: ensure apache is running
        service:
          name: httpd
          state: started
```

```
key_file: /etc/apache2/ssl/mywebsite.key
cert_file: /etc/apache2/ssl/mywebsite.cert
server_name: www.mywebsite.com
```

https://www.middlewareinventory.com/blog/ansible-playbook-example/

Example Ansible Playbook to Setup LAMP stack

```
---
- name: Setting up LAMP Website
  user: vagrant
  hosts: testserver
  become: yes
  tasks:
    - name: latest version of all required packages installed
      yum:
        name:
          - firewalld
          - httpd
          - mariadb-server
          - php
          - php-mysql
        state: latest

    - name: firewalld enabled and running
      service:
        name: firewalld
        enabled: true
        state: started

    - name: firewalld permits http service
      firewalld:
        service: http
        permanent: true
        state: enabled
        immediate: yes

    - name: Copy mime.types file
      copy:
        src: /etc/mime.types
        dest: /etc/httpd/conf/mime.types
        remote_src: yes

    - name: httpd enabled and running
      service:
        name: httpd
        enabled: true
        state: started

    - name: mariadb enabled and running
```

```
    service:
      name: mariadb
      enabled: true
      state: started

  - name: copy the php page from remote using get_url
    get_url:
      url: "https://www.middlewareinventory.com/index.php"
      dest: /var/www/html/index.php
      mode: 0644

  - name: test the webpage/website we have setup
    uri:
      url: http://{{ansible_hostname}}/index.php
      status_code: 200
```

https://www.middlewareinventory.com/blog/ansible-playbook-example/

Exemples:

1. Archive module examples – Ansible
2. Ansible Unarchive module examples
3. Shell module examples
4. Ansible Copy module examples
5. Ansible + Vagrant Playbook for provisioning Apache
6. Ansible Copy SSH Keys between remote servers example
7. How to copy files between remote hosts with ansible
8. Ansible changed_when and failed_when in playbook
9. Ansible Command Module Playbook examples
10. How to use GIT with Ansible playbook
11. Template module examples – Ansible
12. Lookup module examples – Ansible
13. How to read and process JSON file with ansible example
14. Ansible apt module examples
15. Ansible Find module examples
16. How to Process JSON Data with JSON_Query ansible
17. Ansible AWS EC2 Example playbook
18. Ansible async Poll examples
19. How to download file from URL with ansible playbook
20. Ansible lineinfile – How to add, replace, update line in file with ansible
21. Ansible replace module – how to replace texts in ansible
22. How to wait_for task to be completed in playbook example
23. Add users to EC2 instances with SSH Access – Ansible automated
24. Weblogic JMS Queue Creation with Ansible and WLST
25. Ansible inventory_hostname and ansible_hostname example playbook
26. Ansible FirewallD Example Playbook
27. Ansible How to connect using Bastion host
28. Ansible Playbook to find EC2 instances using EFS file system
29. Ansible Playbook to Delete OLD log files in Windows – Ansible Windows
30. Playbook to Find and Replace Default HTML in IIS – Ansible Windows
31. Ansible Windows Example – How to use Ansible with Windows
32. Ansible Select Attr Example – How to Filter Dictionary and Select Items
33. Ansible Map Function Examples – How to Filter List and Dictionaries
34. Ansible Split Function Examples – With String, List and Dictionary
35. Ansible S3 module Examples – How to use ansible aws_s3 module
36. Ansible Copy files Local to Remote
37. Ansible PRE tasks and POST Tasks Introduction and Examples
38. Ansible List Examples – How to create and append items to list
39. Ansible playbook to install KAFKA On Ubuntu
40. Ansible Slack example – How to send Slack notifications from Ansible
41. Ansible Retry example – How to retry an ansible task until the Condition is met
42. Find multiple files with patterns and replace them with Ansible
43. Playbook to install Apache Tomcat
44. How to wait for URL to respond with Ansible URI – Web and API automation
45. Ansible URI module examples – for Web and API automation