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.

Ansible concepts:

Control node
The machine from which you run the Ansible CLI tools (ansible-playbook , ansible, ansible-vault and others). You can use any computer that meets the software requirements as a control node - laptops, shared desktops, and servers can all run Ansible. Multiple control nodes are possible, but Ansible itself does not coordinate across them, see AAP for such features.

Managed nodes
Also referred to as 'hosts', these are the target devices (servers, network appliances or any computer) you aim to manage with Ansible. Ansible is not normally installed on managed nodes, unless you are using ansible-pull, but this is rare and not the recommended setup.

Inventory
A list of managed nodes provided by one or more 'inventory sources'. Your inventory can specify information specific to each node, like IP address. It is also used for assigning groups, that both allow for node selection in the Play and bulk variable assignment. To learn more about inventory, see the Working with Inventory section. Sometimes an inventory source file is also referred to as a 'hostfile'.

Playbooks
They contain Plays (which are the basic unit of Ansible execution). This is both an 'execution concept' and how we describe the files on which ansible-playbook operates. Playbooks are written in YAML and are easy to read, write, share and understand. To learn more about playbooks, see Ansible playbooks.

Plays
The main context for Ansible execution, this playbook object maps managed nodes (hosts) to tasks. The Play contains variables, roles and an ordered lists of

```
tasks and can be run repeatedly. It basically consists of an implicit loop over
the mapped hosts and tasks and defines how to iterate over them.

Roles
A limited distribution of reusable Ansible content (tasks, handlers, variables,
plugins, templates and files) for use inside of a Play. To use any Role resource,
the Role itself must be imported into the Play.

Tasks
The definition of an 'action' to be applied to the managed host. Tasks must
always be contained in a Play, directly or indirectly (Role, or imported/included
task list file). You can execute a single task once with an ad hoc command using
ansible or ansible-console (both create a virtual Play).

Handlers
A special form of a Task, that only executes when notified by a previous task
which resulted in a 'changed' status.

Modules
The code or binaries that Ansible copies to and executes on each managed node
(when needed) to accomplish the action defined in each Task. Each module has a
particular use, from administering users on a specific type of database to
managing VLAN interfaces on a specific type of network device. You can invoke a
single module with a task, or invoke several different modules in a playbook.
Ansible modules are grouped in collections. For an idea of how many collections
Ansible includes, see the Collection Index.

Plugins
Pieces of code that expand Ansible's core capabilities, they can control how you
connect to a managed node (connection plugins), manipulate data (filter plugins)
and even control what is displayed in the console (callback plugins). See Working
with plugins for details.

Collections
A format in which Ansible content is distributed that can contain playbooks,
roles, modules, and plugins. You can install and use collections through Ansible
Galaxy. To learn more about collections, see Using Ansible collections.
Collection resources can be used independently and discretely from each other.

AAP
Short for 'Ansible Automation Platform'. This is a product that includes
enterprise level features and integrates many tools of the Ansible ecosystem:
ansible-core, awx, galaxyNG, and so on.
```

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"va
      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 authorized_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`.
```

nota convé tenir el fingerprint dels hosts destí al known_hosts, es pot fer per exemple amb:

```
$ ssh-keyscan <ip> <ip> <ip> <ip> >> ~/.ssh/known_hosts
```

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"
}
```

Usar un lab amb xarxa privada

Implementa un servidor i tres workers tots ells Debian 11 en la xarxa privada 172.30.30.0/24.

Vagrentfile

```
# -*- 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 "private_network", ip: "172.30.30.10"
    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 "private_network", ip: "172.30.30.#{i+10}"
      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
```

```
$ cat inventory_lab_private.yaml
servernet:
  hosts:
    server:
      ansible_host: 172.30.30.10

workersnet:
  hosts:
    worker1:
      ansible_host: 172.30.30.11
    worker2:
```

```
      ansible_host: 172.30.30.12
    worker3:
      ansible_host: 172.30.30.13

vagrantlabpublic:
  children:
    servernet:
    workersnet:
```

```
$ ansible-inventory -i inventory_lab_private.yaml --list
{
    "_meta": {
        "hostvars": {
            "server": {
                "ansible_host": "172.30.30.10"
            },
            "worker1": {
                "ansible_host": "172.30.30.11"
            },
            "worker2": {
                "ansible_host": "172.30.30.12"
            },
            "worker3": {
                "ansible_host": "172.30.30.13"
            }
        }

…
```

```
$ ssh-keyscan 172.30.30.10 172.30.30.11 172.30.30.12 172.30.30.13 >>
~/.ssh/known_hosts
```

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

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

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

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

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
```

Playbook:

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

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

Altres

```
- name: Ensure docker is at the latest version
  ansible.builtin.apt:
    name: docker
    state: latest
- name: Ensure that docker is started
  ansible.builtin.service:
    name: docker
    state: started
```

Pràctiques

Pràctica Aula

Documentació:

- [ ] https://gitlab.com/manelmellado/ubnt-at-inf
- [ ] https://gitlab.com/edtasixm14/m14-projectes

Implementar un playbook que aplicat a un host de l'aula o a una màquina virtual Vagrant amb xarxa bridge realitzi tota la post instal·lació necessaria per configurar-lo com un host de l'aula del domini de l'escola, amb autenticació d'usuaris LDAP.

01 Post-install Aula: pas a pas com un script

Engegar la màquina vagrant en mode bridge i iniciar sessió dins per poder anar seguint i aplicant les evolucions en el playbook.

Vagtantfile

```
$ cat Vagrantfile
# -*- 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 "private_network", ip: "192.168.56.10"
    srv.vm.network "public_network", bridge: "enp1s0"
    srv.vm.provision "shell", inline: $install_ssh_pubkey_script_debian

  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/
sudo sh -c 'echo "ansible ALL=(ALL) NOPASSWD: ALL" >> /etc/sudoers.d/ansible'
SCRIPT
```

Iniciar sessió vagrant i anotar l'adreça IP. En l'exemple del llista usa una adreça IP privada, comentar / descomentar segons correspongui.

```
$ vagrant ssh
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 85995sec preferred_lft 85995sec
    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 08:00:27:0f:31:47 brd ff:ff:ff:ff:ff:ff
    altname enp0s8
    inet 192.168.56.10/24 brd 192.168.56.255 scope global eth1
     valid_lft forever preferred_lft forever
    inet6 fe80::a00:27ff:fe0f:3147/64 scope link
       valid_lft forever preferred_lft forever
```

playbook.yml

```
---
- name: Post Install Aula (F2G)
  hosts: servernet
  become: true
  tasks:
    - name: Print message
      ansible.builtin.debug:
        msg: Installing

    - name: Install packages
      ansible.builtin.apt:
        state: latest
        pkg:
        - tree
        - nmap
        - vim
        - vim-gtk3
        - mc
        - geany
        - aptitude
        - git
        - openssh-server
        - krb5-user
        - krb5-multidev
        - libpam-mount
        - sssd
        - nfs-common
        - autofs
        - ufw
        - curl
        - gpm
        - pwgen
        update_cache: yes

    - name: Remove packages
      ansible.builtin.apt:
        name: gnome-games
        state: absent

    - name: Remove dependencies that are no longer required
      ansible.builtin.apt:
        autoremove: yes

    - name: Copy file sssd.conf
      ansible.builtin.copy:
        src: ./sssd.conf
        dest: /etc/sssd
        owner: root
        group: root
        mode: '0600'
      ignore_errors: true

    - name: Copy file sudoers.d inf
      ansible.builtin.copy:
        src: ./inf
        dest: /etc/sudoers.d
        owner: root
        group: root
      ignore_errors: true

    - name: Pam Auth Update
      ansible.builtin.command: pam-auth-update --enable mkhomedir sss systemd pwquality unix libpam-mount

    - name: Disable services
      ansible.builtin.service:
        name: "{{ item }}"
        enabled: no
      loop: [ sssd-autofs.socket, sssd-nss.socket ]

    - name: Disable services
      ansible.builtin.service:
        name: "{{ item }}"
        enabled: yes
      loop: [ sssd, rpc-gssd ]

    - name: Add contrib and nonfree to all repos (sed -i '/^[^#]/  s/main$/main contrib non-free/' /etc/apt/sources.list)
      ansible.builtin.apt_repository:
        filename: inf
        repo: "{{ item }}"
        state: present
      loop:
        - deb https://deb.debian.org/debian bullseye contrib non-free
        - deb https://security.debian.org/debian-security bullseye-security contrib non-free
        - deb https://deb.debian.org/debian bullseye-updates contrib non-free
        - deb https://deb.debian.org/debian bullseye-backports contrib non-free

    - name: Install packages (contrib non-free)
      ansible.builtin.apt:
```

```
      pkg: [ chromium, vlc, gimp, tig, meld, gnome-shell-extension-desktop-icons, terminator,
             firmware-realtek, firmware-misc-nonfree, hunspell-ca, hunspell-es ]
      update_cache: true

  - name: Install kernel headers
    ansible.builtin.apt:
      name: linux-headers-{{ ansible_kernel }}
      state: latest

  - name: Hold apt-get and aptitude authomatic updates linux-headers and linux-images
    ansible.builtin.shell: "{{ item }}"
    loop:
    - apt-mark hold linux-image-$(uname -r)
    - apt-mark hold linux-headers-$(uname -r)
    - aptitude hold linux-image-$(uname -r)
    - aptitude hold linux-headers-$(uname -r)
    #- sed -i 's/\#WaylandEnable=false/WaylandEnable=false/' /etc/gdm3/daemon.conf

  - name: Replace  WaylandEnable
    ansible.builtin.replace:
      path: /etc/hosts
      regexp: '#WaylandEnable=false'
      replace: 'WaylandEnable=false'

  # ------------------------------------------------------------------------
  # Mètode (1)
  - name: Test if exists epoptes profe line in /etc/hosts
    ansible.builtin.shell: grep -c "10.200.247.246 profeF2G.informatica.escoladeltreball.org server" /etc/hosts || true
    register: test_grep

  - name: add epoptes profe host server to /etc/hosts
    ansible.builtin.lineinfile:
      dest: /etc/hosts
      line: 10.200.247.246 profeF2G.informatica.escoladeltreball.org server
    when: test_grep.stdout == "0"

  # ------------------------------------------------------------------------
  # Mètode (2)
  - name: Check if epoptes profe host exits in /etc/hosts
    lineinfile:
      state: absent
      path: /etc/hosts
      regexp: '10.200.247.246 profeF2G.informatica.escoladeltreball.org server'
    check_mode: true
    changed_when: false
    register: check

  - name: if line not exits add it to /etc/hosts
    lineinfile:
      state: present
      path: /etc/hosts
      line: 10.200.247.246 profeF2G.informatica.escoladeltreball.org server
    when: check.found == 0
  # ------------------------------------------------------------------------

  - name: Install epoptes client
    ansible.builtin.apt:
      name: epoptes-client
      install_recommends: yes

  - name: Start epoptes and get server key
    ansible.builtin.shell:  epoptes-client -c  profeF2G.informatica.escoladeltreball.org
    ignore_errors: true
```

Executar el playbook i observar les task on hi ha canvis d'estat i on no i en quins casos s'ignora si el resultat falla.

```
$ ansible-playbook -u ansible --key-file ansible_key -i inventory_privat.yaml playbook.yaml

PLAY [Post Install Aula (F2G)]
*************************************************************************************
**

TASK [Gathering Facts]
*************************************************************************************
**
ok: [server]

TASK [Print message]
*************************************************************************************
**
ok: [server] => {
    "msg": "Installing"
}

TASK [Install packages]
*************************************************************************************
**
changed: [server]

TASK [Remove packages]
*************************************************************************************
**
ok: [server]

TASK [Remove dependencies that are no longer required]
*************************************************************************************
ok: [server]

TASK [Copy file sssd.conf]
*************************************************************************************
**
changed: [server]
```

```
TASK [Copy file sudoers.d inf]
*********************************************************************************
**
changed: [server]

TASK [Pam Auth Update]
*********************************************************************************
**
changed: [server]

TASK [Disable services]
*********************************************************************************
**
changed: [server] => (item=sssd-autofs.socket)
changed: [server] => (item=sssd-nss.socket)

TASK [Disable services]
*********************************************************************************
**
ok: [server] => (item=sssd)
ok: [server] => (item=rpc-gssd)

TASK [Add contrib and nonfree to all repos (sed -i '/^[^#]/  s/main$/main contrib
non-free/' /etc/apt/sources.list)] ********************
changed: [server] => (item=deb https://deb.debian.org/debian bullseye contrib non-free)
changed: [server] => (item=deb https://security.debian.org/debian-security bullseye-security contrib non-free)
changed: [server] => (item=deb https://deb.debian.org/debian bullseye-updates contrib non-free)
changed: [server] => (item=deb https://deb.debian.org/debian bullseye-backports contrib non-free)

TASK [Install kernel headers]
*********************************************************************************
**
changed: [server]

TASK [Hold apt-get and aptitude authomatic updates linux-headers and linux-images]
*************************************************
changed: [server] => (item=apt-mark hold linux-image-$(uname -r))
changed: [server] => (item=apt-mark hold linux-headers-$(uname -r))
changed: [server] => (item=aptitude hold linux-image-$(uname -r))
changed: [server] => (item=aptitude hold linux-headers-$(uname -r))

TASK [Replace  WaylandEnable]
*********************************************************************************
**
ok: [server]

TASK [Test if exists epoptes profe line in /etc/hosts]
*************************************************************************
changed: [server]

TASK [add epoptes profe host server to /etc/hosts]
*************************************************************************
changed: [server]

TASK [Check if epoptes profe host exits in /etc/hosts]
*************************************************************************
ok: [server]

TASK [if line not exits add it to /etc/hosts]
*********************************************************************************
**
skipping: [server]

TASK [Install epoptes client]
*********************************************************************************
*****************
changed: [server]

TASK [Start epoptes and get server key]
*********************************************************************************
**
fatal: [server]: FAILED! => {"changed": true, "cmd": "epoptes-client -c  profeF2G.informatica.escoladeltreball.org",
"delta": "0:02:09.842922", "end": "2023-02-23 16:07:26.147096", "msg": "non-zero return code", "rc": 1, "start":
"2023-02-23 16:05:16.304174", "stderr": "140304346826048:error:0200206E:system library:connect:Connection timed
out:../crypto/bio/b_sock2.c:110:\n140304346826048:error:2008A067:BIO routines:BIO_connect:connect
error:../crypto/bio/b_sock2.c:111:\nconnect:errno=110\nepoptes-client ERROR: Failed to fetch certificate from
profeF2G.informatica.escoladeltreball.org:789", "stderr_lines": ["140304346826048:error:0200206E:system
```

```
library:connect:Connection timed out:../crypto/bio/b_sock2.c:110:", "140304346826048:error:2008A067:BIO
routines:BIO_connect:connect error:../crypto/bio/b_sock2.c:111:", "connect:errno=110", "epoptes-client ERROR: Failed to
fetch certificate from profeF2G.informatica.escoladeltreball.org:789"], "stdout": "", "stdout_lines": []}
...ignoring

PLAY RECAP
*********************************************************************************************
************************************
server     : ok=20    changed=13    unreachable=0    failed=0    skipped=1    rescued=0    ignored=1
```

Descripció del playbook per parts

Task 01

Iniciar un nou playbook amb "---", indicar que actua sobre els hosts de srevernet (un sol servidor) i que la connexió ssh ha d'esdevenir amb sudo un usuari privilegiat.

Aquesta primera tasca simplement és un missatge de debug indicant que es procedeix a la instal·lació. Utilitza el mòdul "ansible.builtin.debug" que permet mostrar missatges en el procés de debug que es realitza mentre fa l'aplicació del playbook.

```
---
- name: Post Install Aula (F2G)
  hosts: servernet
  become: true
  tasks:
    - name: Print message
      ansible.builtin.debug:
        msg: Installing
```

Task 02

Actualitzar amb el mòdul "ansible.builtin.apt" la informació dels repositoris amb l'opció "*update_cache*" que és equivalent al "*apt-get update*".

Instal·lar un conjunt de paquets definits en format de llista extesa de YAML. L'opció "*state=latest*" assegura que s'instal·la la darrera versió possible. Si s'indica "*state=present*" simplement s'assegura que el paquet està instal·lat, però no que és la última versió.

```
  - name: Install packages
    ansible.builtin.apt:
      state: latest
      pkg:
      - tree
      - nmap
      - vim
      - vim-gtk3
      - mc
      - geany
      - aptitude
      - git
      - openssh-server
      - krb5-user
      - krb5-multidev
      - libpam-mount
```

```
      - sssd
      - nfs-common
      - autofs
      - ufw
      - curl
      - gpm
      - pwgen
      update_cache: yes
```

Task 03

Eliminar els paquets amb el mòdul apt indicant que l'estat desitjat és "*state=absent*". Si no estan instal·lats no s'instal·la i si ho estan s'eliminen.

```
- name: Remove packages
  ansible.builtin.apt:
    name: gnome-games
    state: absent
```

Task 04

Aplicar l'opció del mòdul apt equivalent a realitzar l'ordre "*apt-get autoremove*" que elimina tots aquells paquets del sistema innecessàris.

```
- name: Remove dependencies that are no longer required
  ansible.builtin.apt:
    autoremove: yes
```

Task 05

Copiar un fitxer de configuració que està al host (extern) al guest destí (en aquest cas a la màquina virtual vagrant). Es copia el fitxer de configuració sssd.conf que configura els serveis d'autenticació.

S'utilitza el mòdul "ansible.builtin.copy" que permet indicar:

- src: la ubicació relativa o absoluta del fitxer al host (exterior).
- dest: la ubicació absoluta del fitxer en el host remot.
- owner: el propietri al que s'assigna el fitxer.
- group: el grup al que s'assigna el fitxer.
- mode: els permisos que s'assignen al fitxer.

```
- name: Copy file sssd.conf
  ansible.builtin.copy:
    src: ./sssd.conf
    dest: /etc/sssd
    owner: root
    group: root
    mode: '0600'
  ignore_errors: true
```

S'ha afegit l'opció "*ignore_errors:true*" perquè si en el moment d'aplicar el playbook no existeix localment en el host el fitxer de configuració no es pari el playbook i es continuin executant les tasques següents.

Task 07

Igual que en la tasca anterior en aquesta tasca es copia un fitxer de configuració extern del host a la màquina remota. En aquest cas es tracta del fitxerd e configuració de sudo.

```
    - name: Copy file sudoers.d inf
      ansible.builtin.copy:
        src: ./inf
        dest: /etc/sudoers.d
        owner: root
        group: root
      ignore_errors: true
```

Task 08

Executar la comanda externa pam-auth-update que realitza la configuració dels mòduls PAM. En especial en aquest desplegament s'utilitza per assegurar-se d'activar el pam_mkhomedir.so que crea el home de l'usuari si no existeix. Les altres opcions indiquen els mecanismes d'autenticació a implementar.

Es passa l mòdul "*ansible.builtin.command*" directament tota l'ordre a executar.

```
    - name: Pam Auth Update
      ansible.builtin.command: pam-auth-update --enable mkhomedir sss systemd
pwquality unix libpam-mount
```

Task 09

Desactiva un conjunt de serveis aplicant el mòdul de gestió de serveis "ansible.builtin.service". Amb l'opció "enabled: no" s'assegura de fer el "systemctl disable" dels serveis indicats.

En aquesta task en el lloc corresponent a indicar el nom del servei hi ha l'element "{{ item }}" que és un placeholder per a cada un dels elements indicats a sota. Utilitzar "loop:" iterarà per a cada un dels noms que formen la llista compacta de YAML que conté els serveis a desabilitar.

Aquesta és la manera de definir un conjunt d'elements (la llista pot ser explícita o compacta) sobre els que aplicar una acció.

```
    - name: Disable services
      ansible.builtin.service:
        name: "{{ item }}"
        enabled: no
      loop: [ sssd-autofs.socket, sssd-nss.socket ]
```

Task 10

Usant el mòdul service s'itera per els serveis que es volen habilitar. És l'equivalent a fer "systemctl enable" dels serveis.

```
- name: Enable services
  ansible.builtin.service:
    name: "{{ item }}"
    enabled: yes
  loop: [ sssd, rpc-gssd ]
```

Task 11

Configurar els repositoris estàndard de debian de apt per tal d'incorporar a part de main també contrib i non-free.

Hi ha diverses maneres de fer-ho, una d'elels és editar el fitxer /etc/apt/aources.list i usant mòduls de processament de text com per exemple ansible.builtin.lineinfile o ansible.builtin.replace o ansible.builtin.template i modificar les línies ja existents afegint-hi les dues paraules requerides. Aquest mecanisme implica treballar amb patrons d'expressió reular.

En aquesta task s'ha optat per un mecanisme més senzill que és generar de nou les línies però només per a contrib i non-free (observar que no hi ha main). Per defecte es generen en un nou fitxer dins del directori de subconfiguracions /etc/apt/sources.list.d/<nom-auto-generat>. En aquest cas el nom del fitxer s'ha especificat que sigui inf.

Així, doncs, hi haurà el fitxer amb els repositoris estàndard provinents de la instal·lació del Debian (/etc/apt/sources.list) que contenen main, i s'afegirà el fitxer /etc/apt/sources.list.d/inf que conté els mateixos repositori (les url) però amb contrib i non-free. El mòdul usat és el "ansible.builtin.repository" que permet definir repositoris apt.

```
- name: Add contrib and nonfree to all repos (sed -i '/^[^#]/ s/main$/main contrib non-free/' /etc/apt/sources.list)
  ansible.builtin.apt_repository:
    filename: inf
    repo: "{{ item }}"
    state: present
  loop:
    - deb https://deb.debian.org/debian bullseye contrib non-free
    - deb https://security.debian.org/debian-security bullseye-security contrib non-free
    - deb https://deb.debian.org/debian bullseye-updates contrib non-free
    - deb https://deb.debian.org/debian bullseye-backports contrib non-free
```

Task 12

Aquesta tasca instala de nou paquets amb apt. No s'han instal·lat abans perquè requerien de la configuració anterior per usar els repositoris amb contrib i non-free.

Abans de la instal·lació cal fer l'equivalent al "apt-get update" per rellegir dels repositoris que es fa amb l'opció "update_cache: true".

```
    - name: Install packages (contrib non-free)
      ansible.builtin.apt:
        pkg: [ chromium, vlc, gimp, tig, meld,
gnome-shell-extension-desktop-icons,
               terminator, firmware-realtek, firmware-misc-nonfree, hunspell-ca,
               hunspell-es ]
        update_cache: true
```

Task 13

Aquesta tasca actualitza amb apt els headers del kernel (necessaris per exemple per Virtualbox). Per indicar la versió de kernel no es fa explícitament sinó que s'utilitza la variable {{ ansible_kernel }} que serà substituïda pel valor apropiat proporcionat per ansible.

```
    - name: Install kernel headers
      ansible.builtin.apt:
        name: linux-headers-{{ ansible_kernel }}
        state: latest
```

Task 14

Aquest tasca executa un conjunt d'ordres shell usant el mòdul "ansible.builtin.shell". Cada una de les ordres a usar esta en la llista i s'hi itera a través de {{ item }} i loop:.

S'utilitza aquest mòdul i no el mòdul *ansible.builtin.command* perquè les ordres a realitzar contenen expansions. Quan cal usar expansions, command substitution, redireccions, etc el mòdul command no serveix i cal usar el mòdul shell.

La finalitat d'aquesta tasca és marcar els paquets del kernel i dels headers perquè no s'actualitzin automàticament.

```
    - name: Hold apt-get and aptitude automatic updates
      ansible.builtin.shell: "{{ item }}"
      loop:
      - apt-mark hold linux-image-$(uname -r)
      - apt-mark hold linux-headers-$(uname -r)
      - aptitude hold linux-image-$(uname -r)
      - aptitude hold linux-headers-$(uname -r)
```

Task 15

Aquesta tasca desactiva Wayland de manera que la pantalla d'inici passarà a usar Xorg. Fa ús del mòdul "ansible.builtin.replace" que permet aplicar recerca i substitució similar a com ho fa l'ordre de sistema operatiu sed.

```
    - name: Replace  WaylandEnable
      ansible.builtin.replace:
        path: /etc/hosts
        regexp: '#WaylandEnable=false'
        replace: 'WaylandEnable=false'
```

Part del playbook corresponent a epoptes

Les tasques finals del playbook instal·len i configuren el client de epoptes. Es fa dues vegades usant dos mètode diferents, simplement a mode de demostració de mecanismes alternatius per fer la mateixa tasca.

Tot i això aquesta és una primera aproximació simplista i rude. Caldria aplicar vairables i templates per fer-ho més professionalment.

Els dos mètodes diferents són per com es configura el fitxer /etc/hosts per tal d'incorprar la resolució de nom del servidor epoptes de l'aula.

Atenció: Cal tenir present que s'ha de buscar una manera d'afegir una entrada al /etc/hosts que sigui idempotent. És a dir, no podem fer simplement un echo "text" >> /etc/hosts perquè cada vegada que s'apliqués el palybook generaria una nova línia.

Exemple incorrecte:

```
    # Atenció, aquesta tasca NO és idempotent!
    #- name: Add Profe host to the /etc/hosts
    #  ansible.builtin.shell: echo "10.200.247.246
profeF2G.informatica.escoladeltreball.org server" >> /etc/hosts
```

Mòdel (1)

Task 16

Per resoldre el problema de l'actualització del /etc/hosts s'utilitzen dues tasques conjuntes. Aquesta primera executa amb shell una ordre grep per buscar si ja existeix o no l'entrada al fitxer /etc/hosts. El resultat del grep queda enregistrar a la variable test-grep usant "register: test_grep".

Observar que l'execució d'una ordre al shell com grep retorna un status d'execució de correcte si troba el text i d'error si no el troba. Però no es vol que un error (si el text no existeix) pari el playbook, per això s'emmascara el valor de retorn amb un "|| true" que garanteix que l'ordre executada amb shell sempre es considera *ok* i no *fail*.

```
    - name: Test if exists epoptes profe line in /etc/hosts
      ansible.builtin.shell: grep -c "10.200.247.246
profeF2G.informatica.escoladeltreball.org server" /etc/hosts || true
      register: test_grep
```

Task 17

Aquesta tasca només s'executa quan la tasca anterir ha desat a la variable "test_grep" el valor 0, que indica que no ha trobat la línia buscada en el fitxer /etc/hosts. Aquesta condició s'estableix amb l'opció "when".

Amb when s'indica quan cal aplicar el mòdul, n aquest cas el mòdul "ansible.builtin.lineinfile" que és un mòdul que permet gestionar text, línies de text, en fitxers. En concret afegeix la línia indicada al fitxer /etc/hosts.

```
- name: add epoptes profe host server to /etc/hosts
  ansible.builtin.lineinfile:
    dest: /etc/hosts
    line: 10.200.247.246 profeF2G.informatica.escoladeltreball.org server
  when: test_grep.stdout == "0"
```

Mètode (2)

Task 18

El procés que es segueix és similar al del mètode anterior: una primera tasca s'encarrega de registrar en una variable si la línia buscada es troba o no dins del fitxer /etc/hosts. Una segona tasca que s'activa només si la línia no hi és, llavors l'afegeix.

Aquest tasca usa el mòdul "ansible.builtin.lineinfile" amb l'opció "state=absent" i "check_mode: true" que valida si la línia està present o no en el fitxer /etc/hosts.

El resultat del check es desa a la variable "check" amb l'opció "register: check", que servira de referència a la propera tasca per saber si cal afegir o no la línia.

Igual que passava en el mètode anterior la línia pot sre-hi o no i per tant l'execució d'aquesta tasca pot generar *ok* i *fail*, però no es vol aturar l'execució del playbook. Per això hi ha l'opció "changed_when: false" que indica que quan la tasca falli en lloc de retornar fail retorni *ok* (una manera més acadèmica de fer el que el mètode anterior feia amb "|| true").

```
- name: Check if epoptes profe host exits in /etc/hosts
  lineinfile:
    state: absent
    path: /etc/hosts
    regexp: '10.200.247.246 profeF2G.informatica.escoladeltreball.org server'
  check_mode: true
  changed_when: false
  register: check
```

Task 19

Si la variable "check" de la tasca anterior conté 0 significa que la línia no era al fitxer /etc/fstab i per tant cal afegir-la. S'afegeix usant el mateix mòdul "ansible.builtin.lineinfile" i ho indica l'opció "state: present".

```
- name: if line not exits add it to /etc/hosts
  lineinfile:
    state: present
    path: /etc/hosts
    line: 10.200.247.246 profeF2G.informatica.escoladeltreball.org server
  when: check.found == 0
```

Task 20

Aquesta tasca instal·la amb apt el paquet del epoptes client utilitzant l'opció "install_recommends".

```
    - name: Install epoptes client
      ansible.builtin.apt:
        name: epoptes-client
        install_recommends: yes
```

Task 21

Aquesta tasca engega el client de epoptes tot obtenint la clau d'accés ssh pública del servidor. S'executa usant el mòdul shell i s'hi indica l'opció "ignore_errors" perquè pot ser que el servidor no estigui apropiadament configurat i no sigui capaç de contactar-hi o d'obtenir-ne la key.

Aquesta tasca segurament trigarà, i més si al final ha de fallar… Però no generarà error en el playbook.

```
    - name: Start epoptes and get server key
      ansible.builtin.shell:  epoptes-client -c
profeF2G.informatica.escoladeltreball.org
      ignore_errors: true
```

Execució del playbook

En l'execució d'un playbbok es pot executar tot de cop o per parts, les opcions més usuals són:

- Llistar les tasques de què es composa
- Executar totes les tasques
- Executar les tasques una a una demanant confirmació.
- Executar les tasques començant a partir d'una tasca determinada.
- Executar les tasques que tenen o no tenen un tag

Llistar les tasques d'un playbook

```
$ ansible-playbook --list-tasks playbook.yaml
[WARNING]: provided hosts list is empty, only localhost is available. Note that
the implicit localhost does not match 'all'
[WARNING]: Could not match supplied host pattern, ignoring: servernet

playbook: playbook.yaml

  play #1 (servernet): Post Install Aula (F2G)     TAGS: []
    tasks:
      Print message   TAGS: []
      Install packages       TAGS: []
      Remove packages TAGS: []
      Remove dependencies that are no longer required     TAGS: []
      Copy file sssd.conf    TAGS: []
      Copy file sudoers.d infTAGS: []
      Pam Auth Update TAGS: []
```

```
      Disable services       TAGS: []
      Enable services TAGS: []
      Add contrib and nonfree to all repos  TAGS: []
      Install packages (contrib non-free)   TAGS: []
      Install kernel headers TAGS: []
      Hold apt-get and aptitude automatic updates linux-headers and linux-images
       TAGS: []
      Replace  WaylandEnable TAGS: []
      Test if exists epoptes profe line in /etc/hosts     TAGS: []
      add epoptes profe host server to /etc/hosts  TAGS: []
      Check if epoptes profe host exits in /etc/hosts     TAGS: []
      if line not exits add it to /etc/hosts       TAGS: []
      Install epoptes client TAGS: []
      Start epoptes and get server key      TAGS: []
```

Executar les tasques una a una demanant confirmació.

```
$ ansible-playbook -u ansible --key-file ansible_key -i inventory_privat.yaml --step
playbook.yaml

PLAY [Post Install Aula (F2G)]
*************************************************************************************
***********************
Perform task: TASK: Gathering Facts (N)o/(y)es/(c)ontinue: y

Perform task: TASK: Gathering Facts (N)o/(y)es/(c)ontinue:
*************************************************************************

TASK [Gathering Facts]
*************************************************************************************
*******************************
ok: [server]
Perform task: TASK: Print message (N)o/(y)es/(c)ontinue: y

Perform task: TASK: Print message (N)o/(y)es/(c)ontinue:
*************************************************************************

TASK [Print message]
*************************************************************************************
*********************************
ok: [server] => {
    "msg": "Installing"
}
Perform task: TASK: Install packages (N)o/(y)es/(c)ontinue:
```

Executar les tasques començant a partir d'una tasca determinada.

```
$ ansible-playbook -u ansible --key-file ansible_key -i inventory_privat.yaml
--start-at-task "Copy file sssd.conf"  playbook.yaml

PLAY [Post Install Aula (F2G)]
*************************************************************************************
***********************

TASK [Gathering Facts]
*************************************************************************************
*******************************
ok: [server]

TASK [Copy file sssd.conf]
*************************************************************************************
**************************
ok: [server]

TASK [Copy file sudoers.d inf]
*************************************************************************************
***********************
ok: [server]

TASK [Pam Auth Update]
*************************************************************************************
*******************************
```

```
changed: [server]

TASK [Disable services]
*********************************************************************************
*****************************
ok: [server] => (item=sssd-autofs.socket)
ok: [server] => (item=sssd-nss.socket)
…
```

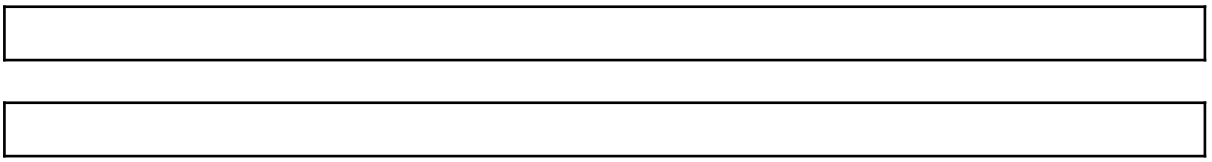
Executar les tasques amb un tag determinat

```
    - name: Test if exists epoptes profe line in /etc/hosts
      ansible.builtin.shell: grep -c "10.200.247.246
profeF2G.informatica.escoladeltreball.org server" /etc/hosts || true
      register: test_grep
      tags:
        - epoptes
        - metode1

    - name: add epoptes profe host server to /etc/hosts
      ansible.builtin.lineinfile:
        dest: /etc/hosts
        line: 10.200.247.246 profeF2G.informatica.escoladeltreball.org server
      when: test_grep.stdout == "0"
      tags:
        - epoptes
        - metode1
```

```
$ ansible-playbook -u ansible --key-file ansible_key -i inventory_privat.yaml -t
epoptes,metode1  playbook.yaml

PLAY [Post Install Aula (F2G)]
*********************************************************************************
************************

TASK [Gathering Facts]
*********************************************************************************
*******************************
ok: [server]

TASK [Test if exists epoptes profe line in /etc/hosts]
*********************************************************************************
changed: [server]

TASK [add epoptes profe host server to /etc/hosts]
*********************************************************************************
****
skipping: [server]

PLAY RECAP
*********************************************************************************
*****************************************
server                     : ok=2    changed=1    unreachable=0    failed=0
skipped=1    rescued=0    ignored=0
```

```
$ ansible-playbook -u ansible --key-file ansible_key -i inventory_privat.yaml
-skip-tags epoptes,metode1  playbook.yaml
```

02 Post-install Aula: usar variables



Pràctica LDAP amb ldapphpadmin (Debian)

Documentació:

- [ ] https://gitlab.com/manelmellado/ubnt-at-inf
- [ ] https://gitlab.com/edtasixm14/m14-projectes

Implementar un playbook que aplicat a un host instal·la el servei LDAP amb la base de dades “edt.org” i el servei phpLdapAdmin per poder visualitzar i administrar gràficament LDAP.

01 Servei LDAP i phpldapadmin

Engegar la màquina vagrant que farà la funció de servidor LDAP. Iniciar sessió dins per poder anar seguint i aplicant les diverses versions del playbook mentre s’està elaborant.

```
$ vagrant up

==> server: Machine 'server' has a post `vagrant up` message. This is a message
==> server: from the creator of the Vagrantfile, and not from Vagrant itself:
==> server:
==> server: Vanilla Debian box. See https://app.vagrantup.com/debian for help and bug
reports

$ vagrant  ssh

vagrant@bullseye:~$ ip a
3: eth1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP group
default qlen 1000
    link/ether 08:00:27:4c:06:d0 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 42713sec preferred_lft 42713sec
    inet6 fe80::a00:27ff:fe4c:6d0/64 scope link
       valid_lft forever preferred_lft forever
```

nota Aquest exemple utilitza el fitxer de playbook que conté els dos plays junts, en format de dos plays.

```
# Si cal verificar permisos de la key
$ chmod 400 ansible_key

$ ansible-playbook -u ansible --key-file ansible_key -i inventory.yaml
playbook_ldap_phpldapadmin_debian.yml
```

Part del play correspondent al servei LDAP

```
PLAY [Install LDAP server with edt.org database]
*********************************************************************************************

TASK [Gathering Facts]
*********************************************************************************************************************
ok: [server]
```

```
TASK [Print message]
*********************************************************************************************************************************
ok: [server] => {
    "msg": "Installing LDAP"
}

TASK [Install LDAP client package]
*********************************************************************************************************************************
ok: [server]

TASK [Install LDAP server package]
*********************************************************************************************************************************
ok: [server]

TASK [Install package utilities]
*********************************************************************************************************************************
ok: [server]

TASK [Recursively remove directories]
*********************************************************************************************************************************
changed: [server] => (item=/etc/ldap/slapd.d)
changed: [server] => (item=/var/lib/ldap)

TASK [Create a directory if it does not exist]
*********************************************************************************************************************************
changed: [server] => (item=/etc/ldap/slapd.d)
changed: [server] => (item=/var/lib/ldap)

TASK [Copy the slapd.conf file]
*********************************************************************************************************************************
ok: [server]

TASK [Copy the edt.org file]
*********************************************************************************************************************************
ok: [server]

TASK [Create dynamic configuration]
*********************************************************************************************************************************
fatal: [server]: FAILED! => {"changed": true, "cmd": ["slaptest", "-f", "/etc/ldap/slapd.conf", "-F",
"/etc/ldap/slapd.d"], "delta": "0:00:00.010190", "end": "2023-02-25 19:03:52.377191", "msg": "non-zero return code", "rc":
1, "start": "2023-02-25 19:03:52.367001", "stderr": "63fa5b98 mdb_db_open: database \"dc=edt,dc=org\" cannot be opened: No
such file or directory (2). Restore from backup!\n63fa5b98 backend_startup_one (type=mdb, suffix=\"dc=edt,dc=org\"):
bi_db_open failed! (2)\nslap_startup failed (test would succeed using the -u switch)", "stderr_lines": ["63fa5b98
mdb_db_open: database \"dc=edt,dc=org\" cannot be opened: No such file or directory (2). Restore from backup!", "63fa5b98
backend_startup_one (type=mdb, suffix=\"dc=edt,dc=org\"): bi_db_open failed! (2)", "slap_startup failed (test would
succeed using the -u switch)"], "stdout": "", "stdout_lines": []}
...ignoring

TASK [Populate ans list]
*********************************************************************************************************************************
changed: [server] => (item=slaptest -f /etc/ldap/slapd.conf -F /etc/ldap/slapd.d -u)
changed: [server] => (item=slapadd  -F /etc/ldap/slapd.d -l /etc/ldap/edt-org.ldif)
changed: [server] => (item=slapcat)

TASK [Recursively change ownership]
*********************************************************************************************************************************
changed: [server] => (item=/etc/ldap/slapd.d)
changed: [server] => (item=/var/lib/ldap)

TASK [Start service httpd, if not started]
*********************************************************************************************************************************
changed: [server]
```

Descripció del playbook: Part LDAP

Task 01

El fitxer comença amb la marca d'inici de fitxer de playbook "---". Es defineix el conjunct de hosts sobre el que actuar i s'indica que cal realitzar les tasques com usuari privilegiat amb "become: true".

Aquest primer fragment mostra la primera tasca consistent en generar un missatge de debug, que es mostren en l'execució del paybook, incicant que s'inicia la instal·lació de LDAP.

```
---
- name: Install LDAP server with edt.org database
  hosts: servernet
  become: true
```

```
tasks:
  - name: Print message
    ansible.builtin.debug:
      msg: Installing LDAP
```

Task 02 03 04

Aquestes tres tasques instalen usant apt els paquets necessaris per LDAP. Estan separades simplement per poder mostrar formats diferents d'utilització del mòdul "ansible.builtin.apt".

```
  - name: Install LDAP client package
    ansible.builtin.apt:
      name: ldap-utils
      state: present
      update_cache: yes

  - name: Install LDAP server package
    ansible.builtin.apt:
      name: slapd
      state: present

  - name: Install package utilities
    ansible.builtin.apt:
      pkg:
      - tree
      - nmap
      - vim
```

Task 05 06

Aquestes dues tasques esborren recursivament el contingut dels directoris de configuració i de base de dades i els generen de nou. Hi ha diverses estratègies per fer això. Aquí s'ha aplicat l'estratègia:

- Primer eliminar els directoris i tot el que contenen usant el mòdul "ansible.builtin.file" que elimina el propi directori indicat i tot el què conté.

- Com que s'han eliminat també els directoris (i no només el contingut que és el que feia falta). Per tant ara es creen de nou aquests directoris.

```
  - name: Recursively remove directories
    ansible.builtin.file:
      path: "{{ item }}"
      state: absent
    loop:
      - /etc/ldap/slapd.d
      - /var/lib/ldap

  - name: Create a directory if it does not exist
    ansible.builtin.file:
      path: "{{ item }}"
      state: directory
      mode: '0755'
    loop:
      - /etc/ldap/slapd.d
      - /var/lib/ldap
```

Una altra estratègia que es podria aplicar és seleccionar tots els elements de dins del directori amb un filtre i eliminar-los iterant per cada element.

```
```

Task 07 08

Copiar el fitxer de configuració i el fitxer de dades a una ubicació dins de la màquina virtual. S'utilitza el mòdul "ansible.builtin.copy".

```
    - name: Copy the slapd.conf file
      ansible.builtin.copy:
        src: ./slapd.conf
        dest: /etc/ldap/slapd.conf

    - name: Copy the edt.org file
      ansible.builtin.copy:
        src: ./edt-org.ldif
        dest: /etc/ldap/edt-org.ldif
```

Task 09 10

Aquestes dues tasques executen ordres de slapd, una per generar la configuració dinàmica i l'altre per fer la càrrega de dades a baix nivell. Com que totes dues ordres no pertanyen a cap mòdul s'executen usant el mòdul "ansible.builtin.command".

La primera d'aquestes dues tasques genera la configuració dinàmica del servidor ldap a partir del fitxer de configuració slapd.conf. Cal destacar que l'ordre slaptest retorna un resultat d'error (ja és normal que ho faci així), però si no s'enmascara el playbook finalitzarà. És per això que s'aplica l'opció "ignore_errors: yes" que permet continuar aplicant les següents tasques del playbook encara que aquesta falli (o digui que falla…).

La segona fa la càrrega o "populate" de la base de dades a baix nivell, abans d'haver engegat el servei. De fet concretament executa tres ordres, la primera repeteix l'slaptest amb l'opció "-u" que verifica que tota la configuració és correcte, la segona carrega les dades a baix nivell i la tercera fa un volcat de les dades inserides a la base de dades ldap.

```
    - name: Create dynamic configuration
      ansible.builtin.command: slaptest -f /etc/ldap/slapd.conf -F
/etc/ldap/slapd.d
      ignore_errors: yes

    - name: Populate ans list
      ansible.builtin.command: "{{ item }}"
      ignore_errors: yes
      loop:
        - slaptest -f /etc/ldap/slapd.conf -F /etc/ldap/slapd.d -u
        - slapadd  -F /etc/ldap/slapd.d -l /etc/ldap/edt-org.ldif
        - slapcat
```

Tasl 11

Abans d'engegar el servei LDAP cal aplicar recursivament al directori de configuració i al directori de dades un canvi de propietari i grup. La instal·lació s'ha fet com a usari privilegiat root i cal que aquests serveis pertanyin a l'usuari ldap (o openldap segons la distribució).

```
- name: Recursively change ownership
  ansible.builtin.file:
    path: "{{ item }}"
    state: directory
    recurse: yes
    owner: openldap
    group: openldap
  loop: [ /etc/ldap/slapd.d, /var/lib/ldap ]
```

Task 12

Aquesta última tasca del play de LDAP engega el servei slapd. De fet aplica l'opció restared per assegurar-se de que es recarrega tota la configuració apropiadament.

```
- name: Start service httpd, if not started
  ansible.builtin.service:
    name: slapd
    state: restarted
```

Descripció del playbook: Part phpLdapAdmin

El playbook que s'està utilitzant en tot asuqst capítol està format per dos plays fiferents en un sol fitxer, un primer play de LDAP i aquest segon de phpLdapAdmin.

```
PLAY [Install phpldapadmin]
*********************************************************************************
***

TASK [Gathering Facts]
*********************************************************************************
***
ok: [server]

TASK [Print message]
*********************************************************************************
***
ok: [server] => {
    "msg": "Installing phpLdapAdmin"
}

TASK [Install packages and dependencies]
*********************************************************************************
***
ok: [server]

TASK [Install a .deb package from the internet]
*********************************************************************************
***
ok: [server]

TASK [Copy the config.php file]
*********************************************************************************
```

```
***
ok: [server]

TASK [Copy the phpldapadmin.conf file]
*********************************************************************************************
***
ok: [server]

TASK [Create dynamic configuration]
*********************************************************************************************
***
changed: [server]

TASK [Start service slapd, if not started]
*********************************************************************************************
***
changed: [server]

PLAY RECAP
*********************************************************************************************
***
server   : ok=21   changed=8    unreachable=0    failed=0    skipped=0    rescued=0    ignored=1
```

Task 13

Aquest bloc indica que es comença un nou play que s'aplica als hosts de la xarxa servernet actuant com l'usuari privilegiat per defecte (que és root).

La tasca que realitza és mostrar un missatge de debug indicant que comença la instal·lació de phpLdapAdmin.

```
- name: Install phpldapadmin
  hosts: servernet
  become: true
  tasks:
    - name: Print message
      ansible.builtin.debug:
        msg: Installing phpLdapAdmin
```

Task 14 15

Aquestes dues tasques instal·len els paquets necessaris per usar phpLdapAdmin i algunes utilitats per verificar-ne el funcionament. En la primera de les tasques es defineix una llista de paquets que s'instal3len amb el mòdul "ansible.builtin.apt".

La segona tasca també instal3la usant el mateix mòdul ansible però el paquet a instal·lar és una URL.

```
    - name: Install packages and dependencies
      ansible.builtin.apt:
        pkg:
        - php-xml
        - apache2
        - tree
        - nmap
        - vim
        update_cache: yes

    - name: Install a .deb package from the internet
```

```
    ansible.builtin.apt:
      deb: http://ftp.de.debian.org/debian/pool/main/p/phpldapadmin/phpldapadmin_1.2.6.3-0.2_all.deb
```

Task 16 17

Ara és el torn de copiar els fitxers de configuració de config.php i phpldapadmin.conf. El primer (config.php) configura el servei i cal assegurar-se que la directiva que indica quin és el servidor ldap a contactar estigui ben definida. El segon (phpldapadmin.conf) configura l'accés a l'aplicació dins del servidor apache2, estableix els permisos d'accés.

Cal recordar que per accedir al servei, un cop engegat cal indicar la URL:
<ip-del-servidor>:port/phpldapadmin

```
  - name: Copy the config.php file
    ansible.builtin.copy:
      src: ./config.php
      dest: /etc/phpldapadmin/config.php

  - name: Copy the phpldapadmin.conf file
    ansible.builtin.copy:
      src: ./phpldapadmin.conf
      dest: /etc/apache2/conf-enabled/phpldapadmin.conf
```

Tal i com s'ha indicat abans és important qu en el fitxer config.php que configura el servei es defineixi correctament com accedir al servei ldap. Dependrà d'on estigui (quin host) el servei LDAP en marxa. En aquest exemple en que tant lDAP com phpLdapAdmin estan en el mateix host cal indicar *localhost*. Si el servidor LDAP es trova en un altre host cal indicar el seu FQDN o l'adreça IP, per exemple *ldap.edt.org*.

Fragment d'exemple de la configuració config.php

```
// $servers->setValue('server','host','127.0.0.1');
$servers->setValue('server','host','localhost');
```

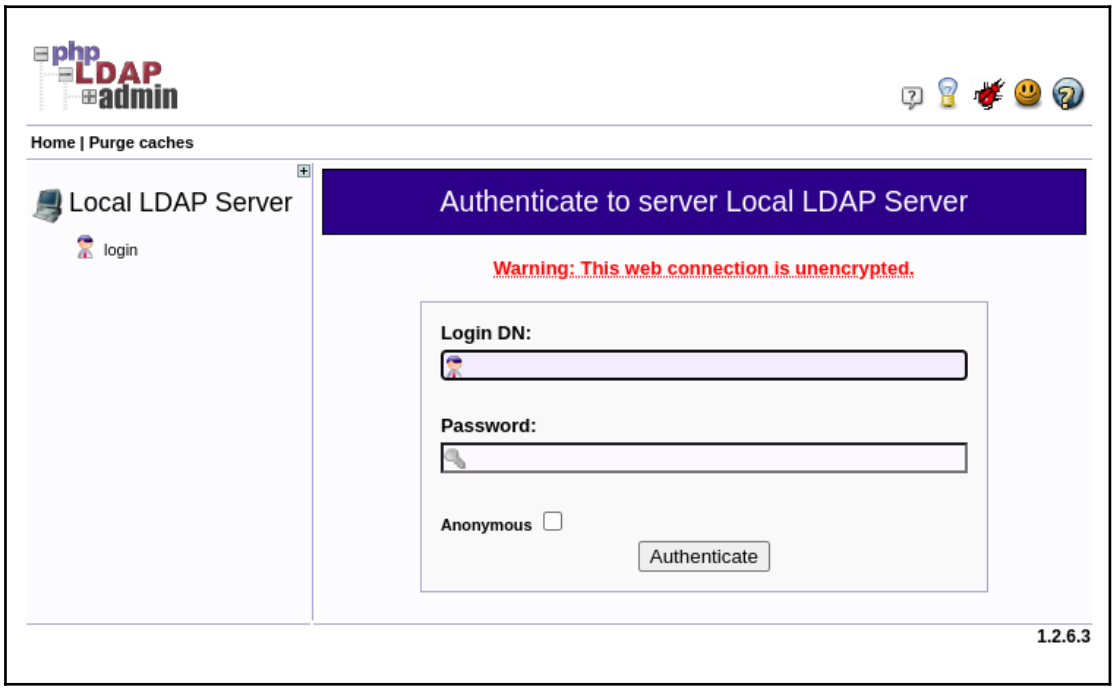
```
// $servers->setValue('server','host','127.0.0.1');
$servers->setValue('server','host','ldap.edt.org');
```

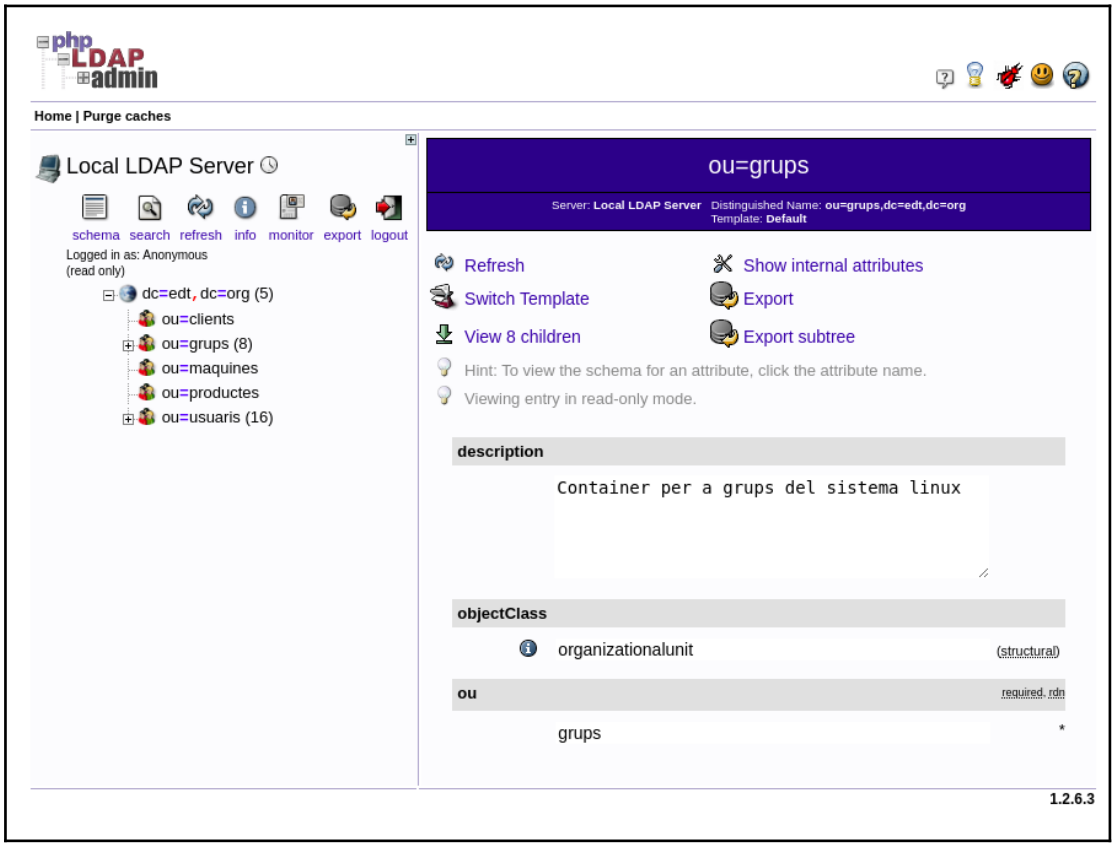
Task 18

Finalment cal engegar el servei apache que activarà la web en php que permet administrar el servei LDAP. S'utilitza el mòdul "ansible.builtin.service".

```
  - name: Start service apache2
    ansible.builtin.service:
      name: apache2
      state: restarted
```

Podem observar que s'ha realitzat la instal3lació apropiadament i que es contacta amb el servidor lDAP.





02 Configuració ansible.cfg i inventory

Anem a definir un fitxer de configuració local que permeti definir l'usuari i el fitxer de clau privada a utilitzar per tal de no haver-ho d'explicitar en la línia d'ordres quan s'executa *ansible-playbook*.

En concret es configurarà:

- Que no es demani el *host_key* en fer la conenxió SSH. Això estalvia les preguntes pessades i el manteniment apropiat del *known_hosts* de SSH. S'utilitza la directiva "host_key_checking".
- Es pot definir també com s'anomena el fitxer d'inventory a tutilitzar de manera que no calgui indicar-lo en cada execució, amb la directtiva "inventory".
- També es pot definir el nom i la ubicació del fitxer amb la clau privada de l'usuari, usant la directiva "private_key_file"
- A vegades es mostren warnings relatius a la versió de l'intèrpret de python en les màquines remotes, amb la directiva "`INTERPRETER_PYTHON = auto_silent`" s'elimina la visualització d'aquests missatges.

ansible.cfg

```
[defaults]
host_key_checking = False
inventory = ./inventory.yaml
private_key_file = ./ansible_key
INTERPRETER_PYTHON = auto_silent
```

També es pot definir en el inventory a nivell global o a nivell particular de cada xarxa o host quin usuari, mètode, password, key, etc usar. En aquest exemple s'indica que l'usuari amb el que connectar és ansible.

inventory.yaml

```
servernet:
  hosts:
    server:
      ansible_host: 192.168.1.39
      ansible_user: ansible
```

Execució sense necessitat d'arguments:

```
$ ansible-playbook  playbook_ldap_phpldapadmin_debian.yml
```

03 Playbook amb includes

En aquest exemple s'ha creat un playbook "main" que té dos plays, el que instal3la LDAp i el que instal·la phpLdapAdmin. Però les tasques de cada cas estan en fitxers a part. S'han generat dos fitxers a part que es carreguen usant la directiva "import_task" o "include_task".

Els fitxers que contenen les tasques no han de tenir la part general que descriu els hosts i el become, sinó que únicament han de contenir les tasks. En fer-ho així aquestes tasques es poden incloure en altres playbooks.

Així en tenir-les per separat es poden incorporar com a fragments sempre que facin falta el playbooks que instal·lin conjunts de coses més grans.

Fitxers amb les tasques per separat:

- LDAP: *part_playbook_ldap_debian.yaml*
- phpLdapAdmin: part_playbook_phpldapadmin_debian.yaml

```
$ cat playbook_main_includes_debian.yaml
---
- name: Install LDAP server with edt.org database
  hosts: servernet
  become: true
  tasks:
    - name: Print message
      ansible.builtin.debug:
        msg: Installing LDAP

    - include_tasks: ./part_playbook_ldap_debian.yaml

- name: Install phpldapadmin
  hosts: servernet
  become: true
  tasks:
    - name: Print message
      ansible.builtin.debug:
        msg: Installing phpLdapAdmin

    - import_tasks: ./part_playbook_phpldapadmin_debian.yaml
```

```
$ head part_playbook_ldap_debian.yaml
    - name: Install LDAP client package
      ansible.builtin.apt:
        name: ldap-utils
        state: present
        update_cache: yes

    - name: Install LDAP server package
      ansible.builtin.apt:
        name: slapd
        state: present
```

```
$ head part_playbook_phpldapadmin_debian.yaml
    - name: Install packages and dependencies
      ansible.builtin.apt:
        pkg:
        - php-xml
        - apache2
        - tree
        - nmap
        - vim
        update_cache: yes

    - name: Install a .deb package from the internet
      ansible.builtin.apt:
        deb: http://ftp.de.debian.org/debian/pool/main/p/phpldapadmin/phpldapadmin_1.2.6.3-0.2_all.deb
```

Execució:

```
$ ansible-playbook  playbook_main_includes_debian.yaml
```

04 Playbook diferenciant entre distribucions GNU/Linux

En aquest exemple s'ha creat un playbook "distro" que aplica tasques diferents segons si el host remot és d'una distribució o altra de GNU/Linux. Permet, doncs, aplicar la mateixa tasca lògica (per exemple instal·lar LDAP) a distribucions diferents aplicant fitxers de tasques diferents segons sigui la distribució.

El mecanisme per executar una tasca segons si es compleix una condició o no és aplicar "when".

Exemple de playbook on es fa include de les tasques d'instal3lació debian quan el sistema operatiu destí és un debian.

```
$ cat playbook_distros_ldap_phpldapadmin.yaml
---
- name: Install LDAP server with edt.org database
  hosts: servernet
  become: true
  tasks:
    - name: Print message
      ansible.builtin.debug:
        msg: Installing LDAP

    - name: Print os information
      ansible.builtin.debug:
        msg: "{{ ansible_os_family }}"

    - include_tasks: ./part_playbook_ldap_debian.yaml
      when: ansible_os_family == 'Debian'

- name: Install phpldapadmin
  hosts: servernet
  become: true
  tasks:
    - name: Print message
      ansible.builtin.debug:
        msg: Installing phpLdapAdmin

    - import_tasks: ./part_playbook_phpldapadmin_debian.yaml
      when: ansible_os_family == "Debian"
```

```
$ ansible-playbook  playbook_distros_ldap_phpldapadmin.yaml

PLAY [Install LDAP server with edt.org database]
*********************************************************************************
**

TASK [Gathering Facts]
*********************************************************************************
**
ok: [server]

TASK [Print message]
*********************************************************************************
**
ok: [server] => {
    "msg": "Installing LDAP"
}

TASK [Print os information]
*********************************************************************************
**
ok: [server] => {
    "msg": "Debian"
}
```

```
TASK [include_tasks]
*********************************************************************************
**
included:
/home/images/ansible/ansible_practica_ldap_php_ldap_admin/part_playbook_ldap_debi
an.yaml for server
…
```

Si s'observa el fitxer de playbook i la documentació es veu que hi ha diversos mètodes per fer els includes:

- Include_task
- Import_task
- Include

Pràctica amb LDAP i phpldapadmin (Alpine)

Documentació:

- ☐ https://gitlab.com/manelmellado/ubnt-at-inf
- ☐ https://gitlab.com/edtasixm14/m14-projectes

Implementar un playbook que aplicat a un host instal·la el servei LDAP amb la base de dades “edt.org” i el servei phpLdapAdmin per poder visualitzar i administrar gràficament LDAP. El host utilitza Alpine GNU/Linux.

```
ansible-galaxy collection install community.general
```

```
$ ansible-galaxy collection install community.general
Process install dependency map
Starting collection install process
Installing 'community.general:6.3.0' to
'/home/ecanet/.ansible/collections/ansible_collections/community/general'
```

Servei LDAP i phpldapadmin

Engegar la màquina vagrant que farà la funció de servidor LDAP utilitzant un Alpine Linux.. Iniciar sessió dins per poder anar practicat i observant les proves fetes amb el playbook.

nota l’exemple usa l’usuari vagrant i el private_key que s’obte de la ruta de vagrant .vagrant/machines/server/virtualbox/private_key.

- Engegar el servidor alpine amb vagrant
- Observar l’adreça ip assignada (és un host usant una interfície bridge que rep una adreça IP pública de la xarxa de l’amfitrió).
- Copiar el fitxer private_key corresponent a la clau provada ssh de vagrant.
- Executar el playbook

```
$ vagrant up

alpine38:~$ ip a
3: eth1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP
qlen 1000
    link/ether 08:00:27:2c:02:eb brd ff:ff:ff:ff:ff:ff
    inet 192.168.1.33/24 scope global eth1
       valid_lft forever preferred_lft forever
    inet6 fe80::a00:27ff:fe2c:2eb/64 scope link
       valid_lft forever preferred_lft forever
```

```
$ cat inventory.yaml
servernet:
  hosts:
    server:
```

```
      ansible_host: 192.168.1.33
      ansible_user: vagrant
```

```
$ cp .vagrant/machines/server/virtualbox/private_key .
```

```
$ ansible-playbook    playbook_distros_ldap_phpldapadmin.yaml
```

Play part corresponent a LDAP

- Observem que el playbook està dividit en dos plays, mostra dos cops el misatge indicamt PLAY, un per instal·lar LDAP i un per instal3lar phpldapadmin.

- Mostra el missatge informat que procedeix a fer un include amb “include_tasks” del fitxer de tasques d’instal3lació de LDAP.

- Observem el missatge de debug que indica quin sistema operatiu ha trobat en el host remot, en aquest cas Alpine.

```
PLAY [Install LDAP server with edt.org database]
************************************************************

TASK [Gathering Facts]
********************************************************************************
*
ok: [server]

TASK [Print message]
********************************************************************************
*
ok: [server] => {
    "msg": "Installing LDAP"
}

TASK [Print os information]
********************************************************************************
*
ok: [server] => {
    "msg": "Alpine"
}

TASK [include_tasks]
********************************************************************************
*
included: /home/images/ansible/ansible_practica_ldap_phpldapadmin_alpine/part_playbook_ldap_alpine.yaml for
server

TASK [Print message]
********************************************************************************
*
ok: [server] => {
    "msg": "Installing LDAP"
}

TASK [Install LDAP client package]
********************************************************************************
ok: [server]

TASK [Recursively remove directories]
************************************************************************
changed: [server] => (item=/etc/ldap/slapd.d)
changed: [server] => (item=/var/lib/openldap)
```

```
TASK [Create a directory if it does not exist]
************************************************************
changed: [server] => (item=/etc/ldap/slapd.d)
changed: [server] => (item=/var/lib/openldap)

TASK [Copy the slapd.conf file]
*****************************************************************************
ok: [server]

TASK [Copy the edt.org file]
*****************************************************************************
**
ok: [server]

TASK [Populate and list]
*****************************************************************************
**
changed: [server]

TASK [Recursively change ownership]
*********************************************************************
changed: [server] => (item=/etc/ldap/slapd.d)
changed: [server] => (item=/var/lib/openldap)

TASK [Start service slapd, if not started]
*******************************************************************
changed: [server]

TASK [include_tasks]
*****************************************************************************
**
skipping: [server]
```

Play part corresponent a phpldapadmin

- Mostra el segon missatge de pLAY informant que comença un nou play.
- Altre cop es mostra quin és el sistema operatiu detectat, un Alpine, i passa a executar les tasques per aquest sistema remot.
- Observar aquí no s'indica la incorporació del fitxer extern de tasques per instal·lar phpldapadmin, s'h ausat l'opció "import_tasks".

```
PLAY [Install phpldapadmin]
*****************************************************************************

TASK [Gathering Facts]
*****************************************************************************
*
ok: [server]

TASK [Print message]
*****************************************************************************
*
ok: [server] => {
    "msg": "Installing phpLdapAdmin"
}

TASK [Print os information]
*****************************************************************************
***
ok: [server] => {
    "msg": "Alpine"
}

TASK [Print message]
```

```
*********************************************************************************
ok: [server] => {
    "msg": "Installing phpLdapAdmin"
}

TASK [Install php apache and phpldapadmin packages]
************************************************************
ok: [server]

TASK [Copy the config.php file]
*********************************************************************************
ok: [server]

TASK [Copy the phpldapadmin.conf file]
*************************************************************************
ok: [server]

TASK [Recursively change ownership]
*************************************************************************
ok: [server]

TASK [Start service apache2]
*********************************************************************************
**
changed: [server]

TASK [Install packages and dependencies]
*********************************************************************
skipping: [server]

TASK [Install a .deb package from the internet]
************************************************************
skipping: [server]

TASK [Copy the config.php file]
*********************************************************************************
skipping: [server]

TASK [Copy the phpldapadmin.conf file]
*************************************************************************
skipping: [server]

TASK [Start service apache2, if not started]
*********************************************************************
skipping: [server]

PLAY RECAP
*********************************************************************************
***
server : ok=22   changed=6    unreachable=0    failed=0    skipped=6    rescued=0    ignored=0
```

Descripció del playbook

Aquest playbook conté dos plays, un per ldap i un per phpldapadmin. Cada un d'ells defineix les característiques necessaries per a la seva execució i fan un include de les tasques necessaries. Aquest include és condicional segons si el sistema remot és Debian o Alpine, per cada cas crida els fitxers apropiats.

Característiques a descriure a cada play:

- Nom del play
- Hosta sobre els que actuar
- Become per executar les tasques com usuari privilegiat
- vars_files per indicar els fitxers de variables a usar.

```
---
- name: Install LDAP server with edt.org database
  hosts: servernet
  become: true
  vars_files:
    - vars_alpine.yaml
…
```

```
- name: Install phpldapadmin
  hosts: servernet
  become: true
  vars_files:
    - vars_alpine.yaml
…
```

Tots dos plays utilitzen el mateix fitxer de variables (es podia haver repartir en dos). Les variables no són compostes, tenen un nom tipus “alpine_ldap_packages” però també es podien haver estructurat com un direccionari compost.

```
$ cat vars_alpine.yaml
alpine_ldap_packages: [ tree, nmap, vim, openldap, openldap-back-mdb,
openldap-back-monitor ]
alpine_ldap_directories: [ /etc/ldap/slapd.d, /var/lib/openldap ]
alpine_ldap_user: ldap
alpine_ldap_slapd:
  source: ./slapd_alpine.conf
  destination: /etc/openldap/slapd.conf
alpine_ldap_ldif:
  source: ./edt-org.ldif
  destination:  /etc/openldap/edt-org.ldif
alpine_phpldapadmin_packages: [ apache2, php5-apache2, phpldapadmin ]
alpine_phpldapadmin_config:
  source: ./config.php
  destination: /etc/phpldapadmin/config.php
alpine_phpldapadmin_phpldapadmin_conf:
  source: ./phpldapadmin_alpine.conf
  destination: /etc/apache2/conf.d/phpldapadmin.conf
alpine_phpldapadmin_user: apache
```

nota si feu més d’un play en un fitxer playbook recordeu que per cada play cal definir totes les seves opcions necessàries, no us descuideu les variables!.

Task 01

La primera tasca del play de ldap és simplemement mostra un missatge de debug informant que es procedeix a la instal·lació de LDAP.

```
   - name: Print message
     ansible.builtin.debug:
       msg: Installing LDAP
```

Task 02 03 04

La primera d’aquestes tres tasques és un ‘xivato’ per veure a través del debug quin és el sistema operatiu detectact en el host remot, utilitza la variable predefinida “ansible_os_family”.

Els dos tasks següents carreguen el fitxer de tasques per instal·lar LDAP amb un include. S'utilitza "include_task" per carregar aquest fitxer de tasques (que no és un playbook per si mateix).

Per seleccionar quan s'ha d'aplicar el fitxer de tasques Debian o el d'apine s'utilitza l'avaluació de la condició "`ansible_os_family == 'Alpine'`" usant l'operador "when".

```
  - name: Print os information
    ansible.builtin.debug:
      msg: "{{ ansible_os_family }}"

  - include_tasks: ./part_playbook_ldap_alpine.yaml
    when: ansible_os_family == 'Alpine'

  - include_tasks: ./part_playbook_ldap_debian.yaml
    when: ansible_os_family == 'Debian'
```

Task 05

Primera tasca del segon play encarregat d'instal·lar phpldapadmin. Simplement mostra pel debug el missatge informat que es procedeix a fer aquesta instal·lació.

```
tasks:
  - name: Print message
    ansible.builtin.debug:
      msg: Installing phpLdapAdmin
```

Task 06 07 08

Igual que s'ha vist amb lDAP aquestes tres tasques actuen conjuntament. La primera mostra quin és el tipus de sistema operatiu detectat en el host remot (posat simplement per poder-ne fer el següiment).

Les altres dues carreguen el fitxer de tasques apropiat segons sigui el destí un sistema debian o un sistema Alpine. Per fer-ho utilitza l'opció "import_task".

```
  - name: Print os information
    ansible.builtin.debug:
      msg: "{{ ansible_os_family }}"

  - import_tasks: ./part_playbook_phpldapadmin_alpine.yaml
    when: ansible_os_family == "Alpine"

  - import_tasks: ./part_playbook_phpldapadmin_debian.yaml
    when: ansible_os_family == "Debian"
```

Utilització de variables

Aquest exemple d'instal3lar LDAP i phpldapadmin a Alpine utilitza variables. S'ha vist que s'ha creat un fitxer de variables anomenat vars_alpine.yaml que conté la definició de totes les variables usades.

Anem a veure dis dels fitxers parcials de tasques com s'han utilitzat aquestes variables.

part_playbook_ldap_alpine.yaml

```
- name: Print message
  ansible.builtin.debug:
    msg: Installing LDAP

- name: Install LDAP client package
  community.general.apk:
    name: "{{ alpine_ldap_packages }}"
    state: latest
    update_cache: yes

- name: Recursively remove directories
  ansible.builtin.file:
    path: "{{ item }}"
    state: absent
  loop: "{{ alpine_ldap_directories }}"

- name: Create a directory if it does not exist
  ansible.builtin.file:
    path: "{{ item }}"
    state: directory
    mode: '0755'
  loop: "{{ alpine_ldap_directories }}"

- name: Copy the slapd.conf file
  ansible.builtin.copy:
    src: "{{ alpine_ldap_slapd.source }}"
    dest: "{{ alpine_ldap_slapd.destination }}"

- name: Copy the edt.org file
  ansible.builtin.copy:
    src: "{{ alpine_ldap_ldif.source }}"
    dest: "{{ alpine_ldap_ldif.destination }}"

- name: Populate and list
  ansible.builtin.command: slapadd  -l "{{ alpine_ldap_ldif.destination }}"
  ignore_errors: yes

- name: Recursively change ownership
  ansible.builtin.file:
    path: "{{ item }}"
    state: directory
    recurse: yes
    owner: "{{ alpine_ldap_user }}"
    group: "{{ alpine_ldap_user }}"
  loop: "{{ alpine_ldap_directories }}"

- name: Start service slapd, if not started
  ansible.builtin.service:
    name: slapd
    state: restarted
```

En el següent exemple es pot veure que la llista de paquets a instal·lar no s'indica directament en el playbook sinó que resideix en una variable anomenada "*alpine_ldap_packages*". A sota es pot observar la definició d'aquesta variable que hi ha al fitxer de "*vars_files*".

```
- name: Install LDAP client package
  community.general.apk:
    name: "{{ alpine_ldap_packages }}"
    state: latest
    update_cache: yes
```

```
alpine_ldap_packages: [ tree, nmap, vim, openldap, openldap-back-mdb,
openldap-back-monitor ]
```

En el següent exemple es pot veure que els directoris a eliminar es proporcionen en forma de llista en el loop, però no s'indiquen directament sinó a través de la variable "*alpine_ldap_directories*". A sota es pot observar la definició d'aquesta variable que hi ha al fitxer de "*vars_files*".

```
    - name: Recursively remove directories
      ansible.builtin.file:
        path: "{{ item }}"
        state: absent
      loop: "{{ alpine_ldap_directories }}"
```

```
alpine_ldap_directories: [ /etc/ldap/slapd.d, /var/lib/openldap ]
```

En el següent exemple es pot veure que el mecanisme usat per copiar fitxers indicant l'origen i el destí és un diccionari anomenat "alpine_ldap_slapd". Cada element consta de dos atributs "source" i "destination". A sota es pot observar la definició d'aquesta variable que hi ha al fitxer de "*vars_files*".

```
    - name: Copy the slapd.conf file
      ansible.builtin.copy:
        src: "{{ alpine_ldap_slapd.source }}"
        dest: "{{ alpine_ldap_slapd.destination }}"
```

```
alpine_ldap_slapd:
  source: ./slapd_alpine.conf
  destination: /etc/openldap/slapd.conf
```

L'exemple següent utilitza també una variable que és un diccionari però veiem que ara s'utilitza dins d'una ordre externa executable amb command.

```
    - name: Populate and list
      ansible.builtin.command: slapadd  -l "{{ alpine_ldap_ldif.destination }}"
      ignore_errors: yes
```

```
alpine_ldap_ldif:
  source: ./edt-org.ldif
  destination:  /etc/openldap/edt-org.ldif
```

L'últim exemple és per assignar l'usuari i grup als fitxers de configuració i dades. En lloc d'assignar-lo directament s'utilitza la variable "alpine_ldap_user" que conté el nom de l'usuari apropiat.

```
    - name: Recursively change ownership
      ansible.builtin.file:
        path: "{{ item }}"
        state: directory
        recurse: yes
        owner: "{{ alpine_ldap_user }}"
        group: "{{ alpine_ldap_user }}"
```

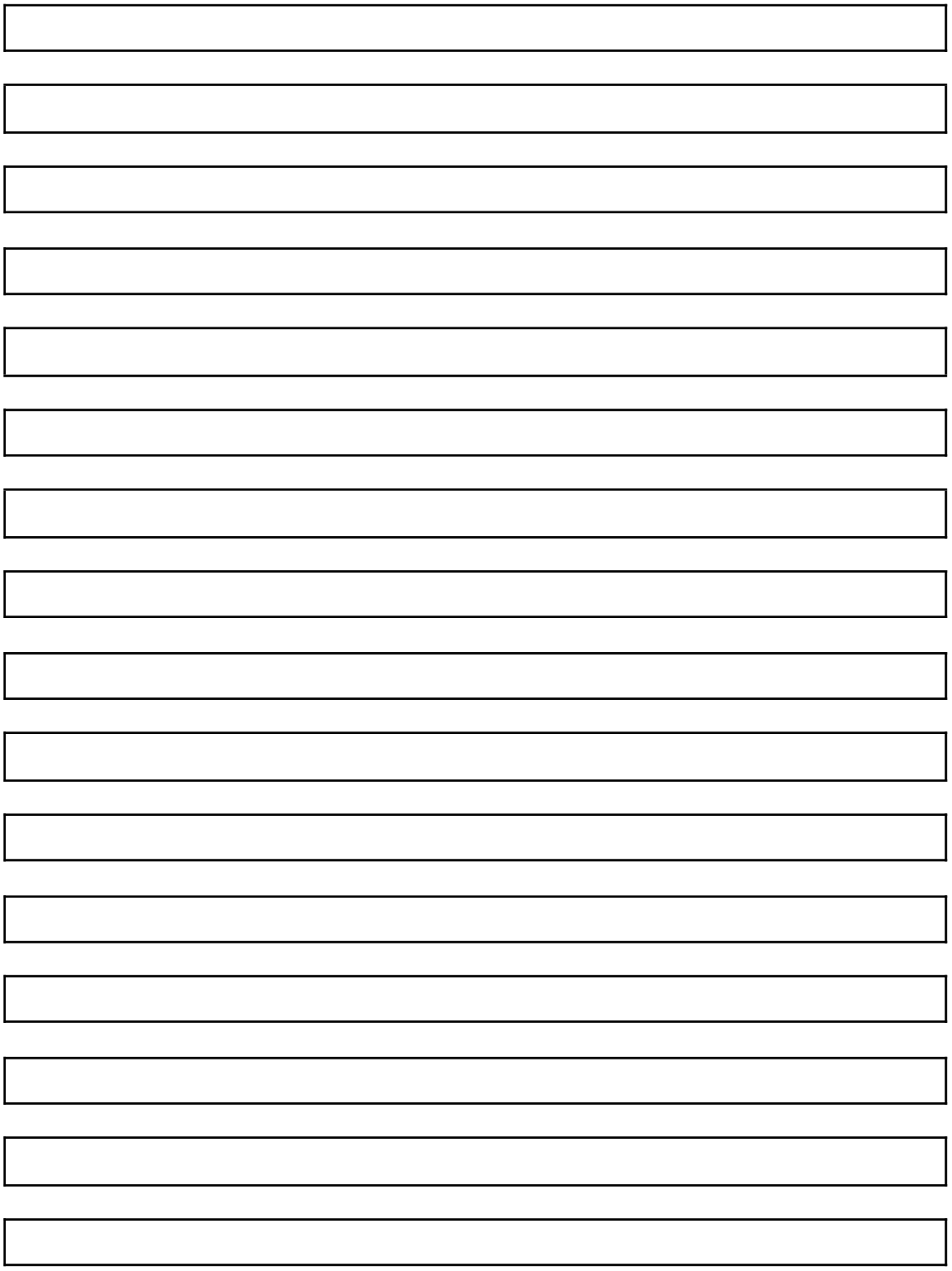
```
alpine_ldap_user: ldap
```

part_playbook_phpldapadmin_alpine_yaml

A continuació es pot observar el fitxer de tasques per insal·lar el servei phpldapadmin en un alpine que utilitza variables en lloc de valors predefinits. Les variables són totes definides al fitxer *vars_alpine.yaml*.

```
- name: Print message
  ansible.builtin.debug:
    msg: Installing phpLdapAdmin

- name: Install php apache and phpldapadmin packages
  community.general.apk:
    name: "{{ alpine_phpldapadmin_packages }}"
    state: latest
    update_cache: yes

- name: Copy the config.php file
  ansible.builtin.copy:
    src: "{{ alpine_phpldapadmin_config.source}}"
    dest: "{{ alpine_phpldapadmin_config.destination }}"

- name: Copy the phpldapadmin.conf file
  ansible.builtin.copy:
    src: "{{ alpine_phpldapadmin_phpldapadmin_conf.source }}"
    dest: "{{ alpine_phpldapadmin_phpldapadmin_conf.destination }}"

- name: Recursively change ownership
  ansible.builtin.file:
    path: "{{ alpine_phpldapadmin_config }}"
    state: directory
    recurse: yes
    owner: "{{ alpine_phpldapadmin_user }}"
    group: "{{ alpine_phpldapadmin_user }}"

- name: Start service apache2
  ansible.builtin.service:
    name: apache2
    state: restarted
```

Pràctica amb distro (alpine + debian) i vars



Ansible up & running

Documentació

Tutorial vídeo sencer:

https://www.youtube.com/watch?v=Wr8zAU-0uR4

Github amb code samples del llibre

https://github.com/ansiblebook/ansiblebook

Book Ansible up & running 3rd edition (part)

https://edu.anarcho-copy.org/GNU%20Linux%20-%20Unix-Like/DevOps/Ansible_Up%20and%20Running%203ED%20ER.pdf
Codi online: https://github.com/ansiblebook

Book Ansible up & running 2th edition pdf

https://www.pdfdrive.com/ansible-up-and-running-e187539038.html https://www.pdfdrive.com/download.pdf?id=187539038&h=54969c2d424d0a8ca3f3839fa3fe2bcf&u=cache&ext=pdf

Ansible up & running 3r edition Video Tutorials

https://www.oreilly.com/library/view/introduction-to-ansible/9781491955956/video245950.html
https://www.oreilly.com/library/view/introduction-to-ansible/9781491955956/
https://www.oreilly.com/library/view/introduction-to-ansible/9781491955956/video245965.html
Són tots compactats en un video de youtube: https://www.youtube.com/watch?v=Wr8zAU-0uR4

Chapter 1 Introduction

Desplegament del lab i iniciació a ansible

- Desplegar els hosts amb Vagrant (server: 172.30.30.10, worker1: 172.30.30.11, worker2: 172.30.30.12 i worker3: 172.30.30.13).
- Verificar l'accés amb l'usuari ansible.
- Configurar l'inventory.
- Test amb ping de l'inventory.
- Configuració d'ansible amb *ansible.cfg*.
- Mòdul per fer ordres del sistema: command.

Engegar el lab de Vagrant

- Engegar el lab, mostrar els hosts actius i observar quina és la clau ssh privada de vagrant per si es vol usar accés a les màquines virtuals amb la clau ssh que genera vagarnt.

```
$ vagrant up

$ vagrant status
Current machine states:

server                    running (virtualbox)
worker1                   running (virtualbox)
worker2                   running (virtualbox)
worker3                   running (virtualbox)

$ vagrant ssh-config server
Host server
  HostName 127.0.0.1
  User vagrant
  Port 2222
  UserKnownHostsFile /dev/null
  StrictHostKeyChecking no
  PasswordAuthentication no
                                                                    IdentityFile
/home/images/ansible/ansible_up_and_running/chapter1/.vagrant/machines/server/virtualbox
/private_key
  IdentitiesOnly yes
  LogLevel FATAL
```

Com que a les nostre smàquines s'ha creat un usuari ansible que pot esdevenir administrador amb sudo, podem accedir-hi per ssh:

```
$ ssh -i ansible_key ansible@172.30.30.10
ansible@bullseye:~$
```

Inventory

Anem a fer diferents tipus d'inventory:

- Inventory general peparat per a fer les pràctiques amb el lab de 4 màquines virtuals en YAML

- El mateix en format INI.
- Inventory només del server amb accés via l'usuari vagrant i la clau privada ssh generada automàticament per vagrant.
- Inventory per accedir només al server usant l'usuari ansible i la clau provada preparada per nosaltres per a l'usuari ansible.

Inventory per al lab de vagrant amb una xarxa servernet amb el server, una workersnet amb els tres workers i una supraxarxa vagrantlab que inclou les altres dues. Aquest inventory és en format YAML.

```
$ cat inventory_lab_private.yaml
servernet:
  hosts:
    server:
      ansible_host: 172.30.30.10

workersnet:
  hosts:
    worker1:
      ansible_host: 172.30.30.11
    worker2:
      ansible_host: 172.30.30.12
    worker3:
      ansible_host: 172.30.30.13
      # per la connexió
      #ansible_user: vagrant
      #ansible_password: vagrant
      #ansible_become: yes

vagrantlab:
  children:
    servernet:
    workersnet:

$ ansible-inventory -i inventory_lab_private.yaml --list
```

Inventory per al lab de vagrant amb una xarxa servernet amb el server, una workersnet amb els tres workers i una supraxarxa vagrantlab que inclou les altres dues. Aquest inventory és en format INI.

```
$ cat inventory_lab_private.ini
[servernet]
172.30.30.10

[workersnet]
172.30.30.11
172.30.30.12
172.30.30.13

[vagrantlab:children]
servernet
workersnet

$ ansible-inventory -i inventory_lab_private.ini --list
```

Inventory per accedir a la VM amb l'usuari vagrant i la clau privada ssh enerada automàticament per vagrant. Accedeix al localhost (no a la ip de la VM) al port 2222 que és on vagrant propaga el port ssh de la màquina virtual. Format INI.

```
$ cat inventory_vagrant.ini
```

```
[webservers]
testserver ansible_port=2222
[webservers:vars]
ansible_host=127.0.0.1
ansible_user = vagrant
ansible_private_key_file = .vagrant/machines/server/virtualbox/private_key

$ ansible-inventory -i inventory_vagrant.ini --list
```

```
$ cat inventory_ansible.ini
[webservers]
testserver ansible_port=22
[webservers:vars]
ansible_host=172.30.30.10
ansible_user = ansible
ansible_private_key_file = ansible_key

$ ansible-inventory -i inventory_ansible.ini --list
```

```
$ cat inventory_aws.ini
[webservers]
testserver ansible_host=ec2-203-0-113-120.compute-1.amazonaws.com
[webservers:vars]
ansible_user=ec2-user
ansible_private_key_file=/path/to/keyfile.pem

$ ansible-inventory -i inventory_aws.ini --list
```

Test amb ping de la inventory

Podem verificar la connectivitat d'ansible amb els hosts que formen part de l'inventory realitzant un ping amb la ordre *ansible* i incicant el mòdul *ping*.

- Podem indicar les options una a una (xarxa, inventori, usuari, clau privada ssh)
- Si la informació ja es troba en el fitxer INI o yAML no cal indicar-la.
- *nota* atenció a la gestió dels fingerprints (claus de host) dels hosts destí del SSH que han d'estar al fitxer *.ssh/known_hosts*.

Exemple fent ping a tot l'inventory yaml

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

Exemple fent ping només a servernet de INI

```
$ ansible servernet -i inventory_lab_private.ini -u ansible --key-file ansible_key -m ping
172.30.30.10 | SUCCESS => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python3"
    },
    "changed": false,
    "ping": "pong"
}
```

Exemple fent ping usant l'usuari vagrant als workers

```
$ ansible webservers -i inventory_vagrant.ini -m ping
testserver | SUCCESS => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python3"
    },
    "changed": false,
    "ping": "pong"
}
```

Exemple usant l'usuari ansible fent ping a tot l'inventory (només té un host)

```
$ ansible all -i inventory_ansible.ini -m ping
testserver | SUCCESS => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python3"
    },
    "changed": false,
    "ping": "pong"
}
```

Configuració d'ansible amb ansible.cfg

Es pot personalitzar la configuració d'ansible i en especial les opcions que es vol evitar posar constantment a la línia d'ordres amb el fitxer *ansible.cfg*. Aquest fitxer pot tenir varis àmbits:

- file specified by the ANSIBLE_CONFIG environment variable
- ./ansible.cfg (ansible.cfg in the current directory)
- ~/.ansible.cfg (.ansible.cfg in your home directory)
- /etc/ansible/ansible.cfg

En el següent exemple es pot observar un fitxer de configuració ansible local al directori de desplegament. Aquest fitxer permet definir el nom de l'inventory a usar de manera que no caldrà passar-lo amb el *-i nom-inventory*. Una altra opció molt interessant és *host_key_checking* que permet indicar al SSH que no faci la ferragossa

```
$ cat ansible.cfg
[defaults]
inventory = inventory_ansible.ini
host_key_checking = false
stdout_callback = yaml
callback_enabled = timer
```

Podem observar com sense indicar l'inventory ha efectuat el ping

```
$ ansible all  -m ping
```

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

Mòdul command

Aquest mòdul *command* permet realitzar ordres en els hosts destí. De fet és el mòdul per defecte que s'utilitza si no s'indica. Per passar quina ordre a fer i amb quines opcions s'utilitza *-a ordre-i-opcions* que és allò que ha de fer.

Mostrar uptime

```
$ ansible all  -m command -a uptime 2> /dev/null
testserver | CHANGED | rc=0 >>
 16:36:47 up 59 min,  1 user,  load average: 0.02, 0.03, 0.01
```

Mostrar uname -a (observar la encapsulació)

```
$ ansible all  -m command -a "uname -a" 2> /dev/null
testserver | CHANGED | rc=0 >>
Linux bullseye 5.10.0-20-amd64 #1 SMP Debian 5.10.158-2 (2022-12-13) x86_64 GNU/Linux
```

Si l'usuari de la connexió (ansible en aquest exemple) no té permisos per realitzar l'ordre, com per exemple usar apt-get, aquesta fallarà.

```
$ ansible all  -m command -a "apt-get update" 2> /dev/null
testserver | FAILED | rc=100 >>
Reading package lists...E: List directory /var/lib/apt/lists/partial is missing. -
Acquire (13: Permission denied)non-zero return code
```

Observem en l'exemple següent que falla l'execució de tree i nmap perquè aquestes ordres no estan instal·lades al sistema.

```
$ ansible all  -m command -a "tree" 2> /dev/null
testserver | FAILED | rc=2 >>
[Errno 2] No such file or directory: b'tree'

$ ansible all  -m command -a "nmap" 2> /dev/null
testserver | FAILED | rc=2 >>
[Errno 2] No such file or directory: b'nmap'
```

Mòdul package

Amb el mòdul *package* es poden realitzar accions relatives a la gestió de packets, instal·lar, desintal·lar, etc. Pot passar, però, que:

- no trobi el paquet perquè no ha actualitzat la lista de repositoris, llavors cal fer prèviament aquesta actualització. En aquest cas cal afegir l'opció *update_cache=yes*.

- No tingui permisos per a fer l'acció perquè necessita ser un usuari privilegiat. En aquest cas cal indicar que cal escalar privilegis. Per poder fer-ho cal usar l'opció *-b* o *--become*.

Exemple on s'intenten instal·lar els paquets tree i nmap i falla

```
$ ansible all  -b -m package -a name=tree
testserver | FAILED! => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python3"
    },
    "changed": false,
    "msg": "No package matching 'tree' is available"
}

$ ansible all  --become -m package -a name=nmap
testserver | FAILED! => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python3"
    },
    "changed": false,
    "msg": "No package matching 'nmap' is available"
}
```

Mòdul service

Aquest mòdul permet governar sreveis, engegar-los, aturar-los, status, show, etc. Per defecte si no s'indica el contrari fa un show.

```
$ ansible all  --become -m service -a name=sshd
testserver | SUCCESS => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python3"
    },
    "changed": false,
    "name": "sshd",
    "status": {
        "ActiveEnterTimestamp": "Thu 2023-02-16 15:37:30 UTC",
        "ActiveEnterTimestampMonotonic": "4181592",
        "ActiveExitTimestamp": "Thu 2023-02-16 15:37:30 UTC",
```

Chapter 2. Playbooks: A Beginning

Una mica de YAML i JSON

Strings

```
YAML
Aquest es un string i no cal encapsular, però es pot si es vol

JSON
“Aquest es un string que cal encapsular en JSON”
```

Booleans

```
YAML
true, True, TRUE, yes, Yes, YES, on, On, ON
false, False, FALSE, no, No, NO, off, Off, OFF

JSON
true
flase
```

Lists

```
YAML
llibres:
  - Totes les bésties de carrega
  - Demà a les tres de la matinada
  - Joc brut

YAML
Llibres: [ Totes les bésties de carrega, Demà a les tres de la matinada, Joc brut ]

JSON
{
 “Llibres”: [
 “Totes les bésties de carrega”,
 “Demà a les tres de la matinada”,
 “Joc brut”
 ]
}

{
"shows": [
"My Fair Lady",
"Oklahoma",
"The Pirates of Penzance"
]
```

```
}
```

Dictionaries

```
YAML
address:
  street: Evergreen Terrace
  appt: '742'
  city: Springfield
  state: North Takoma

YAML
address: { street: Evergreen Terrace, appt: '742', city: Springfield, state: North
Takoma}

JSON
{
  "address": {
  "street": "Evergreen Terrace",
  "appt": 742,
  "city": "Springfield",
  "state": "North Takoma"
  }
}
```

Multi-line strings

https://github.com/lorin/ansible-quickref
https://github.com/ansiblebook/ansiblebook/tree/master/chapter04/playbooks
https://galaxy.ansible.com/

```
[ssh_connection]
ssh_args = -o ControlMaster=auto -o ControlPersist=60s -o ForwardAgent=yes
```

ansible.cfg
[defaults]
host_key_checking = False

```
- name: check out the repository on the host
git: repo={{ repo_url }} dest={{ proj_path }} accept_hostkey=yes
```

```
$ ansible-config init --disabled > ansible.cfg
$ ansible-config init --disabled -t all > ansible.cfg
```

```
[defaults]
inventory = inventory
remote_user = vagrant
private_key_file = ~/.vagrant.d/insecure_private_key
host_key_checking = False
```

```
ansible_host Default
Name of host Description
Hostname or IP address to SSH to
ansible_port 22 Port to SSH to
ansible_user Root User to SSH as
ansible_password (None) Password to use for SSH authentication
ansible_connection smart How Ansible will connect to host (see the following section)
ansible_private_key_file (None) SSH private key to use for SSH authentication
ansible_shell_type sh Shell to use for commands (see the following section)
ansible_python_interpreter /usr/bin/
python Python interpreter on host (see the following section)
ansible_*_interpreter Like ansible_python_interpreter for other languages (see the
following
section)
```

inventory

```
[web]
web[01:20].example.com

[django:children]
web
task

vagrant1 ansible_host=127.0.0.1 ansible_port=2222

[all:vars]
ntp_server=ntp.ubuntu.com

[vagrant:vars]
db_primary_host=vagrant3
db_primary_port=5432
```

inventory

```
db:
user: widgetuser
password: pFmMxcyD;Fc6)6
name: widget_production
primary:
host: rhodeisland.example.com
port: 5432
replica:
host: virginia.example.com
port: 5432
rabbitmq:
host: pennsylvania.example.com
port: 5672
```

```
- name: Provision a vagrant machine
hosts: localhost
vars:
box: trusty64
```

```
tasks:
- name: create a Vagrantfile
command: vagrant init {{ box }} creates=Vagrantfile
- name: Bring up a vagrant machine
command: vagrant up
- name: add the vagrant machine to the inventory
add_host: >
name=vagrant
ansible_host=127.0.0.1
ansible_port=2222
ansible_user=vagrant
ansible_private_key_file=/Users/lorin/.vagrant.d/
insecure_private_key
- name: Do something to the vagrant machine
hosts: vagrant
become: yes
tasks:
# The list of tasks would go here
- ...
```

```
Example 3-13. Creating ad hoc groups based on Linux distribution
- name: group hosts by distribution
hosts: myhosts
gather_facts: True
tasks:
- name: create groups based on distro
group_by: key={{ ansible_distribution }}
- name: do something to Ubuntu hosts
hosts: Ubuntu
tasks:
- name: install htop
apt: name=htop
# ...
- name: do something else to CentOS hosts
hosts: CentOS
tasks:
- name: install htop
yum: name=htop
# ...
```

vars

```
vars:
key_file: /etc/nginx/ssl/nginx.key
cert_file: /etc/nginx/ssl/nginx.crt
conf_file: /etc/nginx/sites-available/default
server_name: localhost
```

```
vars_files:
- nginx.yml
```

```
- debug: var=myvarname
```

```
- name: capture output of whoami command
command: whoami
register: login
```

```
- name: show return value of command module
hosts: server1
tasks:
- name: capture output of id command
command: id -un
```

```
register: login
- debug: var=login
```

```
- name: capture output of id command
command: id -un
register: login
- debug: msg="Logged in as user {{ login.stdout }}"
```

```
- name: print out operating system
hosts: all
gather_facts: True
tasks:
- debug: var=ansible_distribution
```

```
$ ansible server1 -m setup
```

```
Example 4-9. /etc/ansible/facts.d/example.fact
[book]
title=Ansible: Up and Running
author=Lorin Hochstein
publisher=O'Reilly Media

- name: print ansible_local
debug: var=ansible_local
- name: print book title
debug: msg="The title of the book is {{ ansible_local.example.book.title }}"
```

```
Example 4-10. Using set_fact to simplify variable reference
- name: get snapshot id
shell: >
aws ec2 describe-snapshots --filters
Name=tag:Name,Values=my-snapshot
| jq --raw-output ".Snapshots[].SnapshotId"
register: snap_result
- set_fact: snap={{ snap_result.stdout }}
```

```
Table 4-1. Built-in variables
Parameter
hostvars Description
A dict whose keys are Ansible hostnames and values are dicts that map variable names to
values
inventory_hostname Fully qualified domain name of the current host as known by Ansible
(e.g., myhost.exam
ple.com )
inventory_hostname_short Name of the current host as known by Ansible, without the
domain name (e.g., myhost )
group_names A list of all groups that the current host is a member of
groups A dict whose keys are Ansible group names and values are a list of hostnames that
are members
of the group. Includes all and ungrouped groups: {"all": [...], "web": [...],
"ungrouped": [...]}
ansible_check_mode A boolean that is true when running in check mode (see “Check Mode”
on page 313)
ansible_play_batch A list of the inventory hostnames that are active in the current
batch (see “Running on a Batch
of Hosts at a Time” on page 175)
ansible_play_hosts A list of all of the inventory hostnames that are active in the
current play
ansible_version A dict with Ansible version info: {"full": 2.3.1.0", "major": 2,
"minor":
3, "revision": 1, "string": "2.3.1.0"}
```

```
Example 4-11. Setting a variable from the command line
$ ansible-playbook example.yml -e token=12345
```

```
- name: pass a message on the command line
hosts: localhost
vars:
greeting: "you didn't specify a message"
tasks:
- name: output a message
debug: msg="{{ greeting }}"
If we invoke it like this:
$ ansible-playbook greet.yml -e greeting=hiya
```

```
- name: get the current shell
debug: msg="{{ lookup('env', 'SHELL') }}"
```

```
- name: create deploy postgres user
postgresql_user:
name: deploy
password: "{{ lookup('password', 'deploy-password.txt') }}"
```

```
This host runs {{ ansible_distribution }}
```

```
- name: output message from template
debug: msg="{{ lookup('template', 'message.j2') }}"
```

```
lookup('csvfile', 'sue file=users.csv delimiter=, col=1')
```

```
lookup('csvfile', username + ' file=users.csv delimiter=, col=1')
```

```
$ dig +short ansiblebook.com TXT
"isbn=978-1491979808"
```

```
- name: look up TXT record
debug: msg="{{ lookup('dnstxt', 'ansiblebook.com') }}
```

```
Table 8-4. Looping constructs
Name
with_items Input
List
with_lines Command to execute Loop over lines in command output
with_fileglob Glob
Loop over filenames
with_first_found List of paths
First file in input that exists
with_dict Dictionary
Loop over dictionary elements
with_flattened List of lists
Loop over flattened list
with_indexed_items List
Single iteration
with_nested List
Nested loop
with_random_choice List
Single iteration
with_sequence Sequence of integers Loop over sequence
with_subelements List of dictionaries
Nested loop
with_together List of lists
Loop over zipped list
with_inventory_hostnames Host pattern
```

```
Looping strategy
Loop over list elements
Loop over matching hosts
```

```
{
"address": "10.0.2.15",
"netmask": "255.255.255.0",
"network": "10.0.2.0"
}
```

```
- name: iterate over ansible_eth0
debug: msg={{ item.key }}={{ item.value }}
with_dict: "{{ ansible_eth0.ipv4 }}"
```