PRÀCTICA 5:

IaaC

Nom 1: David Morillo Massagué
NIU 1: 1666540

Nom 2: Adrià Muro Gómez
NIU 2: 1665191

Usuari utilitzat a la pràctica: gixpd-ged-22

Taula de continguts

Introducció..... 3
Objectius..... 3
Procediment..... 4
Configuració inicial de l'entorn..... 4
Creació de màquines virtuals amb Terraform..... 5
Configuració terraform provider..... 5
Definició de les VMs..... 5
Configuració del disc..... 6
Configuració dels gràfics..... 6
Configuració de la xarxa..... 7
Configuració del Sistema operatiu..... 7
Template i aprovisionament..... 8
Generació de l'inventari d'Ansible..... 8
Execució de les màquines virtuals..... 9
Aprovisionament de les màquines virtuals amb Ansible..... 12
Estructura del Playbook..... 12
Instal·lació de les dependències..... 12
Instal·lació de Docker..... 13
Instal·lació de Kubernetes..... 13
Instal·lació de Minikube..... 14
Execució del playbook sobre les màquines desplegades..... 15
Validació de la infraestructura..... 17
Conclusions..... 18

Introducció

En aquesta pràctica, ens centrarem en l'ús de tecnologies modernes d'Infraestructura com a Codi (IaaC) per desplegar i gestionar màquines virtuals (MV) a través d'OpenNebula. Utilitzarem Terraform per definir, aprovisionar i configurar les MVs de manera declarativa, i Ansible per automatitzar la instal·lació i configuració de programari necessari.

Aquest enfocament permet una gestió eficient i escalable d'entorns virtualitzats, facilitant tant el desplegament com el manteniment.

Aquest exercici té un doble propòsit: familiaritzar-nos amb eines IaaC i desenvolupar una comprensió més profunda dels sistemes virtuals i de la infraestructura automatitzada.

Objectius

- Implementar un entorn virtualitzat: Crear i gestionar màquines virtuals a OpenNebula utilitzant Terraform.

- Configurar infraestructura automatitzada: Instal·lar i configurar eines necessàries (Docker, kubectl i Minikube) a través de playbooks d'Ansible.

- Generar un inventari dinàmic: Automatitzar la generació de fitxers d'inventari per a l'ús amb Ansible.

- Adquirir habilitats en DevOps: Practicar conceptes fonamentals d'IaaC, incloent-hi aprovisionament automatitzat i configuració post-instal·lació.

Procediment

Configuració inicial de l'entorn

Per començar amb la pràctica, vam configurar l'entorn necessitant les eines Terraform i Ansible per poder gestionar les màquines virtuals i automatitzar la seva configuració.

Descarrega de Terraform: Vam iniciar el procés descarregant i instal·lant Terraform per tal de poder gestionar la infraestructura amb OpenNebula. Per això, utilitzant el comandament wget, vam descarregar la darrera versió (1.10.0) de Terraform des de la pàgina oficial:

```
dakur@lg:~$ wget https://releases.hashicorp.com/terraform/1.10.0/terraform_1.10.0_linux_amd64.zip
--2024-11-29 23:35:57--  https://releases.hashicorp.com/terraform/1.10.0/terraform_1.10.0_linux_amd64.zip
Resolving releases.hashicorp.com (releases.hashicorp.com)... 13.225.47.107, 13.225.47.65, 13.225.47.54, ...
Connecting to releases.hashicorp.com (releases.hashicorp.com)|13.225.47.107|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 27713245 (26M) [application/zip]
Saving to: 'terraform_1.10.0_linux_amd64.zip'
```

Després de descarregar el fitxer ZIP, vam descomprimir-lo amb *unzip terraform_1.10.0_linux_amd64.zip*

Un cop descomprimit, vam moure l'executable terraform a la ruta /usr/local/bin/ per assegurar-nos que fos accessible des de qualsevol lloc amb la comanda *mv terraform /usr/local/bin/*.

D'aquesta manera, vam completar la instal·lació de Terraform i vam verificar que estava correctament instal·lat.

```
adminp@adminp:~/TAX$ terraform -version
Terraform v1.10.0-alpha20240911
on linux_amd64
+ provider registry.terraform.io/hashicorp/local v2.5.2
+ provider registry.terraform.io/opennebula/opennebula v1.4.1
```

Descarrega d'Ansible: Per a l'automatització de la configuració, també vam necessitar Ansible. Així que, per instal·lar-lo en el sistema, vam utilitzar el gestor de paquets apt per instal·lar Ansible des del repositori oficial d'Ubuntu. Primer, vam actualitzar la llista de paquets amb *apt update*.

Seguidament vam instal·lar Ansible amb *apt install ansible*.

```
adminp@adminp:~/TAX$ ansible --version
ansible [core 2.16.3]
  config file = None
  configured module search path = ['/home/adminp/.ansible/plugins/modules', '/usr/share/ansible/plugins/modules']
  ansible python module location = /usr/lib/python3/dist-packages/ansible
  ansible collection location = /home/adminp/.ansible/collections:/usr/share/ansible/collections
  executable location = /usr/bin/ansible
  python version = 3.12.3 (main, Nov  6 2024, 18:32:19) [GCC 13.2.0] (/usr/bin/python3)
  jinja version = 3.1.2
  libyaml = True
```

Amb aquests passos vam tenir Terraform i Ansible instal·lats i configurats correctament, preparant l'entorn per començar amb la creació de màquines virtuals i la seva configuració automatitzada.

Creació de màquines virtuals amb Terraform

Configuració terraform provider

Per començar, vam configurar el proveïdor de Terraform per establir la connexió amb l'API d'OpenNebula. Es va crear un fitxer anomenat providers.tf, on vam especificar el proveïdor necessari i la seva versió (OpenNebula/opennebula).

També vam afegir l'endpoint de l'API d'OpenNebula i les credencials de l'usuari per tal de garantir l'autenticació. Aquesta configuració inicial és imprescindible per poder utilitzar Terraform amb OpenNebula.

```
terraform {
  required_providers {
    opennebula = {
      source  = "OpenNebula/opennebula"
      version = "~> 1.3"
    }
  }
}

provider "opennebula" {
  endpoint = "http://nebulacaos.uab.cat:2633/RPC2"
  username = "gixpd-ged-22"
  password = "9YtyubUY79py"
}
```

Definició de les VMs

A continuació, vam definir les màquines virtuals al fitxer *main.tf*. Es van crear dues màquines virtuals utilitzant el paràmetre *count* de Terraform, que ens va permetre crear múltiples instàncies sense duplicar codi. Els noms de les màquines es van establir com "virtual-machine-0" i "virtual-machine-1".

Es va definir una configuració uniforme per a totes dues, assignant-los 0.5 CPU, 2 vCPUs i 1024 MB de memòria, així com permisos de seguretat configurats com "660". A més, en la descripció de les màquines es va especificar com "Màquina Virtual creada amb Terraform".

```
resource "opennebula_virtual_machine" "vm" {
  count = 2
  name  = "virtual-machine-${count.index}"
  description = "Màquina Virtual creada amb Terraform"
  cpu = 0.5
  vcpu = 2
  memory = 1024
  permissions = "660"
```

Configuració del disc

Posteriorment, vam configurar un disc per a cadascuna de les màquines virtuals. Per això, es va adjuntar un disc amb una mida de 20.000 MB, utilitzant la imatge "Ubu24.04v1.3", de la qual es va recuperar l'ID a través de les dades de la imatge.

```
data "opennebula_image" "ubuntu_image" {
  name = "Ubu24.04v1.3"
}
```

El dispositiu de destinació es va establir com "vda" i es va utilitzar el controlador "qcow2" per assegurar la compatibilitat amb el sistema.

```
disk {
  image_id = data.opennebula_image.ubuntu_image.id
  size = 20000
  target = "vda"
  driver = "qcow2"
}
```

Configuració dels gràfics

Seguint amb la configuració, vam afegir gràfics VNC a les màquines virtuals per tal de facilitar la seva gestió remota.

El tipus de gràfics es va establir com "VNC", i es va habilitar l'escolta a l'adreça "0.0.0.0", permetent connexions des de qualsevol IP. Finalment, es va configurar el mapa de tecles com "es" per adaptar-se al teclat espanyol.

```
graphics {
  type = "VNC"
  listen = "0.0.0.0"
  keymap = "es"
}
```

Configuració de la xarxa

Per connectar les màquines virtuals a la xarxa, vam definir una interfície de xarxa per a cadascuna d'elles. El model de la interfície es va establir com "virtio", utilitzant la xarxa virtual "Internet". També es va associar cada màquina virtual amb el grup de seguretat per defecte anomenat "default".

```
nic {
  network_id = data.opennebula_virtual_network.internet_network.id
  model = "virtio"
  security_groups = [data.opennebula_security_group.default_security_group.id]
}
```

Tant l'ID de la xarxa virtual com l'ID del grup de seguretat es recupera des de les dades de la xarxa i dades del grup de seguretat, respectivament.

```
data "opennebula_virtual_network" "internet_network" {
  name = "Internet"
}

data "opennebula_security_group" "default_security_group" {
  name = "default"
}
```

Configuració del Sistema operatiu

A nivell de sistema operatiu, es va especificar que l'arquitectura de les màquines seria "x86_64" i es va establir que el dispositiu d'arrencada fos "disk0", assegurant així que les màquines utilitzessin el disc configurat prèviament com a font principal d'arrencada.

```
os {
  arch = "x86_64"
  boot = "disk0"
}
```

Template i aprovisionament

Pel que fa a l'ús de plantilles, es va seleccionar la plantilla "Ubuntu 24.04-GIxPD" d'OpenNebula, de la qual es va recuperar l'ID per utilitzar-la.

```
data "opennebula_template" "ubuntu_template" {
  name = "Ubuntu 24.04-GIxPD"
}
```

Addicionalment, es va implementar un aprovisionador local-exec que escaneja l'adreça IP de la màquina virtual i l'afegeix al fitxer /home/adminp/.ssh/known_hosts. Aquesta automatització assegura que es pugui establir una comunicació segura amb les màquines mitjançant SSH.

```
template_id = data.opennebula_template.ubuntu_template.id

provisioner "local-exec" {
    command = <<EOT
    sleep 10
    ssh-keyscan ${self.nic[0].computed_ip} >> /home/adminp/.ssh/known_hosts
    EOT
}
```

Va ser necessari crear una comanda multilínia degut que *ssh-keyscan* s'executava abans de que les IP's de les màquines virtuals estiguessin disponibles.

Per tal d'evitar el problema es va afegir *sleep 10* per donar un temps a les màquines virtuals per computar la seva IP de manera exitosa. *<<EOT, EOT* és la manera d'avisar on comença i on acaba la comanda multilínia.

Generació de l'inventari d'Ansible

Finalment, es va generar un fitxer d'inventari per a Ansible anomenat hosts (dins de la carpeta templates), que inclou totes les adreces IP de les màquines virtuals creades.

Aquest inventari es va generar automàticament mitjançant el recurs local_file de Terraform, utilitzant plantilles de text per donar format al contingut.

```
resource "local_file" "ansible_inventory" {
  content = templatefile("${path.module}/templates/hosts.tpl", {
    vms = opennebula_virtual_machine.vm
  })
  filename = "${path.module}/hosts"
}
```

L'arxiu hosts.tpl va ser el següent:

```
[all:vars]
ansible_connection=ssh
ansible_user=adminp
ansible_ssh_pass=pnimda
ansible_sudo_pass=pnimda

[all]
%{ for vm in vms ~}
${vm.nic[0].computed_ip}
%{ endfor ~}
```

Es van incloure detalls com l'usuari SSH, les credencials i altres paràmetres necessaris per facilitar l'accés i la gestió de les màquines virtuals amb Ansible. També es va crear un bucle que recorre les diferents màquines virtuals agafant les ip's.

Execució de les màquines virtuals

Després de completar la configuració dels fitxers de Terraform, vam procedir a executar els passos necessaris per desplegar l'entorn definit. Primer, vam executar la comanda *terraform init*, que inicialitza el directori de treball i descarrega els proveïdors necessaris, incloent-hi el proveïdor d'OpenNebula.

```
adminp@adminp:~/TAX$ terraform init
Initializing the backend...
Initializing provider plugins...
- Reusing previous version of hashicorp/local from the dependency lock file
- Reusing previous version of opennebula/opennebula from the dependency lock file
- Using previously-installed hashicorp/local v2.5.2
- Using previously-installed opennebula/opennebula v1.4.1

Terraform has been successfully initialized!

You may now begin working with Terraform. Try running "terraform plan" to see
any changes that are required for your infrastructure. All Terraform commands
should now work.

If you ever set or change modules or backend configuration for Terraform,
rerun this command to reinitialize your working directory. If you forget, other
commands will detect it and remind you to do so if necessary.
```

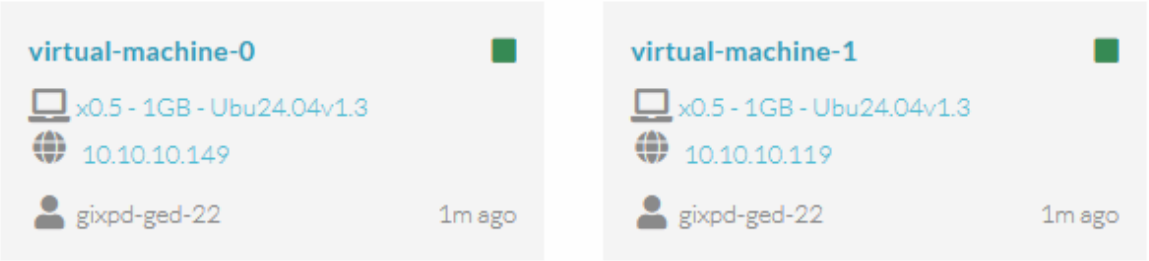
Un cop inicialitzat l'entorn, vam executar *terraform apply*, que aplica els canvis definits al fitxer *main.tf*. Durant l'execució d'aquesta comanda, es va generar un output amb les adreces IP de les màquines virtuals creades, confirmant que s'havien desplegat correctament segons la configuració proporcionada.

```
Apply complete! Resources: 3 added, 0 changed, 0 destroyed.

Outputs:

vm_ips = [
  "10.10.10.149",
  "10.10.10.119",
]
```

Finalment, es va generar l'arxiu d'inventari hosts, que inclou les adreces IP de les màquines creades i permet la seva gestió amb Ansible.

```
[all:vars]
ansible_connection=ssh
ansible_user=adminp
ansible_ssh_pass=pnimda
ansible_sudo_pass=pnimda

[all]
10.10.10.149
10.10.10.119
```

Per verificar que el desplegament s'havia realitzat amb èxit, es va accedir a l'entorn d'OpenNebula, on es podia observar que les dues màquines virtuals apareixien en funcionament, complint amb les especificacions definides.



A més, des de la màquina on s'estava configurant l'entorn, es van realitzar pings a cadascuna de les màquines creades per assegurar la seva connectivitat. Els pings es van completar correctament, confirmant que les màquines estaven operatives i accessibles.

```
adminp@adminp:~/TAX$ ping 10.10.10.149
PING 10.10.10.149 (10.10.10.149) 56(84) bytes of data.
64 bytes from 10.10.10.149: icmp_seq=1 ttl=64 time=0.911 ms
64 bytes from 10.10.10.149: icmp_seq=2 ttl=64 time=0.777 ms
64 bytes from 10.10.10.149: icmp_seq=3 ttl=64 time=0.665 ms
```

```
adminp@adminp:~/TAX$ ping 10.10.10.119
PING 10.10.10.119 (10.10.10.119) 56(84) bytes of data.
64 bytes from 10.10.10.119: icmp_seq=1 ttl=64 time=0.906 ms
64 bytes from 10.10.10.119: icmp_seq=2 ttl=64 time=0.970 ms
64 bytes from 10.10.10.119: icmp_seq=3 ttl=64 time=0.983 ms
```

Aprovisionament de les màquines virtuals amb Ansible

Estructura del Playbook

Per començar, vam crear un playbook d'Ansible anomenat install_tools.yaml. Aquest playbook s'encarrega de gestionar totes les tasques d'instal·lació i configuració de manera automatitzada.

El playbook està configurat per apuntar a tots els amfitrions, habilitar l'escalada de privilegis mitjançant *become: true* i recollir informació dels sistemes objectiu utilitzant *gather_facts: true*. Aquestes configuracions són crucials per assegurar que es puguin executar tasques que requereixen permisos d'administrador i per obtenir informació detallada dels nodes abans de procedir amb les tasques.

```yaml
- name: Configuración d'eines VM
  hosts: all
  become: true
  gather_facts: true
```

Instal·lació de les dependències

Una de les primeres tasques del playbook és "Instal·lar dependències". Aquesta tasca utilitza el mòdul apt per instal·lar un conjunt de paquets essencials per a la configuració posterior. Els paquets instal·lats són:

- apt-transport-https: Permet la comunicació HTTPS per a APT.
- ca-certificates: Assegura la confiança en els certificats HTTPS.
- curl: Una eina necessària per transferir dades.
- software-properties-common: Proporciona utilitats per gestionar repositoris addicionals.

Aquesta tasca està configurada per executar-se amb permisos elevats (become: true), ja que instal·lar paquets al sistema requereix privilegis administratius.

```yaml
tasks:
  - name: Instal·lar dependències
    apt:
      name:
        - apt-transport-https
        - ca-certificates
        - curl
        - software-properties-common
      state: present
      update_cache: yes
```

Instal·lació de Docker

Per instal·lar Docker, vam incloure una sèrie de tasques detallades al playbook d'Ansible.

- Afegir la clau GPG de Docker: La clau GPG es baixa del servidor oficial de Docker i s'afegeix al sistema utilitzant el mòdul apt_key. Aquesta clau és essencial per verificar la integritat dels paquets descarregats del repositori de Docker i assegurar que provenen d'una font fiable. L'URL utilitzat per obtenir la clau és el següent:

```
- name: Afegeix la clau GPG de Docker
  apt_key:
    url: https://download.docker.com/linux/ubuntu/gpg
    state: present
```

- Afegir el repositori de Docker: Mitjançant el mòdul apt_repository, es configura el sistema per utilitzar el repositori oficial de Docker. Això es fa per garantir que s'instal·lin sempre les versions més recents i compatibles del programari.

```
- name: Afegeix el repositori APT de Docker
  apt_repository:
    repo: "deb [arch=amd64] https://download.docker.com/linux/ubuntu focal stable"
    state: present
```

- Instal·lar Docker: Amb el mòdul apt, especificant el paquet docker-ce (Docker Community Edition).

```
- name: Instal·la Docker
  apt:
    name: docker-ce
    state: present
    update_cache: yes
```

Instal·lació de Kubernetes

La instal·lació de Kubernetes es va planificar inicialment utilitzant el repositori alternatiu kubernetes-xenial, una opció habitualment recomanada per instal·lar kubectl en sistemes Ubuntu. Aquest procediment implicava afegir el repositori i actualitzar la memòria cau dels paquets per després instal·lar el paquet kubectl.
No obstant, durant l'execució, l'actualització de la memòria cau dels paquets va fallar amb errors relacionats amb la clau pública GPG i la signatura del repositori.

Aquest problema es va deure al fet que el repositori Xenial estava configurat per versions antigues d'Ubuntu, i no era compatible amb les polítiques de seguretat i configuració de versions modernes com Ubuntu 20.04 o 22.04. Això ens va obligar a canviar de metodologia.

Les tasques per instal·lar Kubernetes van ser les següents:

- Creació del directori per a les claus GPG: Crea el directori /etc/apt/keyrings si no existeix. Aquest directori és utilitzat per emmagatzemar les claus GPG, que són necessàries per verificar la integritat dels paquets del repositori.

```
- name: Crea el directori per a keyrings
  ansible.builtin.file:
    path: /etc/apt/keyrings
    state: directory
    mode: '0755'
```

- Descàrrega de la clau GPG de Kubernetes: Es baixa la clau GPG oficial de Kubernetes des de l'URL: *https://pkgs.k8s.io/core:/stable:/v1.28/deb/Release.key*. Aquesta clau es desa al directori creat anteriorment, amb permisos adequats per assegurar que només es pugui llegir però no modificar.

```
- name: Descarrega la clau GPG de Kubernetes
  ansible.builtin.shell:
    cmd: |
      curl -fsSL https://pkgs.k8s.io/core:/stable:/v1.28/deb/Release.key | gpg --dearmor -o /etc/apt/keyrings/kubernetes-apt-keyring.gpg
    creates: /etc/apt/keyrings/kubernetes-apt-keyring.gpg
```

- Configuració del repositori de Kubernetes: Amb el mòdul apt_repository, es configura el repositori oficial de Kubernetes. Aquesta configuració utilitza la clau descarregada prèviament (/etc/apt/keyrings/kubernetes-apt-keyring.gpg) per assegurar que només es puguin instal·lar paquets signats correctament.

```
- name: Configura el repositori de Kubernetes
  ansible.builtin.shell:
    cmd: |
      echo "deb [signed-by=/etc/apt/keyrings/kubernetes-apt-keyring.gpg] https://pkgs.k8s.io/core:/stable:/v1.28/deb/ /" > /etc/apt/sources.list
    creates: /etc/apt/sources.list.d/kubernetes.list
```

- Instal·lació de Minikube: Finalment, es va instal·lar kubectl utilitzant el mòdul apt. Aquesta eina permet gestionar clústers Kubernetes i és imprescindible per interactuar amb el sistema.

```
- name: Instal·la kubectl
  apt:
    name: kubectl
    state: present
    update_cache: yes
```

Instal·lació de Minikube

- Descàrrega i instal·lació de Minikube: Per instal·lar Minikube es fa ús del mòdul shell d'Ansible per executar una sèrie de comandes en un script bash. La comanda *curl -LO* descarrega l'executable més recent de Minikube des de l'URL oficial, emmagatzemant-lo al directori actual.

Amb la comanda *chmod +x*, es fa que l'arxiu descarregat sigui executable, permetent la seva execució des de la línia de comandes. La comanda *mv* permet moure l'arxiu descarregar a */usr/local/bin*.

Finalment, el paràmetre *args: executable: /bin/bash* assegura que l'script s'executa utilitzant l'intèrpret de comandes Bash, garantint compatibilitat amb les comandes utilitzades.

```
- name: Descarrega i instal·la Minikube
  shell: |
    curl -LO https://storage.googleapis.com/minikube/releases/latest/minikube-linux-amd64
    chmod +x minikube-linux-amd64
    mv minikube-linux-amd64 /usr/local/bin/minikube
  args:
    executable: /bin/bash
```

Execució del playbook sobre les màquines desplegades

Per dur a terme la instal·lació de les eines, vam executar el playbook Ansible anomenat install_tools.yaml utilitzant la comanda següent:

ansible-playbook -i hosts install_tools.yaml

Aquesta comanda s'assegura que el playbook s'executi a tots els amfitrions definits en el fitxer d'inventari hosts, que conté les màquines virtuals configurades anteriorment.

Durant l'execució, Ansible va començar a processar cada tasca definida al playbook, executant-les en les màquines de destinació. A mesura que cada tasca es completava, Ansible va mostrar un resultat de "ok" o "changed" per indicar l'estat de la tasca:

"ok": Aquesta indicació va aparèixer quan la tasca es va executar correctament, sense necessitat de canvis.

"changed": Aquesta indicació va aparèixer quan es va realitzar un canvi en el sistema de destinació, com la instal·lació de paquets o la configuració de repositoris.

```
adminp@adminp:~/TAX$ ansible-playbook -i hosts install_tools.yaml

PLAY [Configuración d'eines VM] ************************************************************************************************************************************

TASK [Gathering Facts] *********************************************************************************************************************************************
ok: [10.10.10.149]
ok: [10.10.10.119]

TASK [Instal·lar dependències] *************************************************************************************************************************************
changed: [10.10.10.119]
changed: [10.10.10.149]

TASK [Afegeix la clau GPG de Docker] *******************************************************************************************************************************
changed: [10.10.10.119]
changed: [10.10.10.149]

TASK [Afegeix el repositori APT de Docker] *************************************************************************************************************************
changed: [10.10.10.149]
changed: [10.10.10.119]

TASK [Instal·la Docker] ********************************************************************************************************************************************
changed: [10.10.10.119]
changed: [10.10.10.149]

TASK [Crea el directori per a keyrings] ****************************************************************************************************************************
ok: [10.10.10.119]
ok: [10.10.10.149]

TASK [Descarrega la clau GPG de Kubernetes] ************************************************************************************************************************
changed: [10.10.10.119]
changed: [10.10.10.149]
```

```
TASK [Ajusta permisos de la clau] **********************************************************************************************************************************
ok: [10.10.10.149]
ok: [10.10.10.119]

TASK [Configura el repositori de Kubernetes] ***********************************************************************************************************************
changed: [10.10.10.149]
changed: [10.10.10.119]

TASK [Actualitza la memòria cau dels paquets] **********************************************************************************************************************
changed: [10.10.10.119]
changed: [10.10.10.149]

TASK [Instal·la kubectl] *******************************************************************************************************************************************
changed: [10.10.10.149]
changed: [10.10.10.119]

TASK [Descarrega i instal·la Minikube] *****************************************************************************************************************************
changed: [10.10.10.119]
changed: [10.10.10.149]

PLAY RECAP *********************************************************************************************************************************************************
10.10.10.119               : ok=12   changed=9    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
10.10.10.149               : ok=12   changed=9    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
```

És important destacar que no es van presentar errors durant l'execució, la qual cosa indica que totes les tasques es van completar correctament en els amfitrions de destinació. Això valida que el playbook es va executar amb èxit i que totes les eines (Docker, Kubernetes i Minikube) es van instal·lar sense cap incidència.

Validació de la infraestructura

Per acabar de confirmar la instal·lació, es va accedir a les dues màquines virtuals creades i es va verificar la versió de Docker, Minikube i Kubernetes (kubectl).

Es va executar les comandes corresponents en cada màquina per assegurar-se que les versions de les eines eren les correctes i que la instal·lació havia estat reeixida.

```
adminp@adminp:~$ docker --version
Docker version 27.3.1, build ce12230
adminp@adminp:~$ minikube version
minikube version: v1.34.0
commit: 210b148df93a80eb872ecbeb7e35281b3c582c61
adminp@adminp:~$ kubectl version
Client Version: v1.28.15
Kustomize Version: v5.0.4-0.20230601165947-6ce0bf390ce3
```

Aquestes comprovacions van confirmar que Docker, Minikube i Kubernetes estaven instal·lats i funcionant correctament.

Conclusions

1. Habilitats apreses:

    - S'ha après a gestionar infraestructures amb Terraform per crear màquines virtuals a OpenNebula.
    - S'ha adquirit experiència en automatitzar la configuració de Docker, Kubernetes i Minikube amb Ansible

2. Problemes trobats:

    - S'han presentat problemes amb les claus GPG i repositoris durant la instal·lació de Kubernetes, que s'han solucionat creant un directori per les claus i actualitzant les configuracions de repositori.
    - Algunes incompatibilitats de versions han requerit l'ajust dels repositoris i l'ús de versions específiques de paquets.

3. Èxits aconseguits:

    - Les màquines virtuals s'han creat amb èxit, i les eines Docker, Kubernetes i Minikube s'han instal·lat correctament.
    - El playbook d'Ansible ha automatitzat amb èxit la instal·lació i configuració de totes les eines, facilitant el procés.