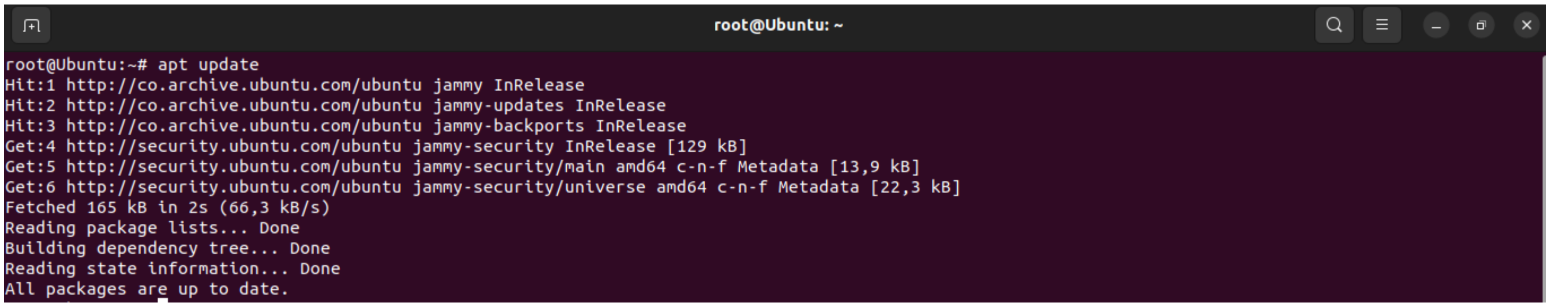
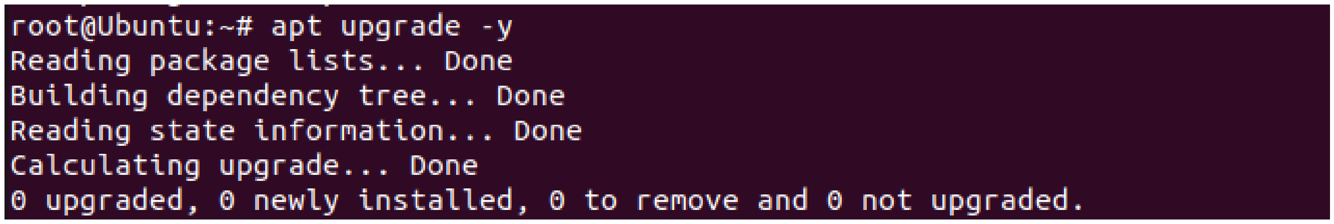
Instalación de Minikube o K3s en Linux, Distribución Grafica: Ubuntu

Opción A – Minikube (estar como root)

1. **Actualizar sistema (Comandos Basicos: apt update, apt upgrade -y)**



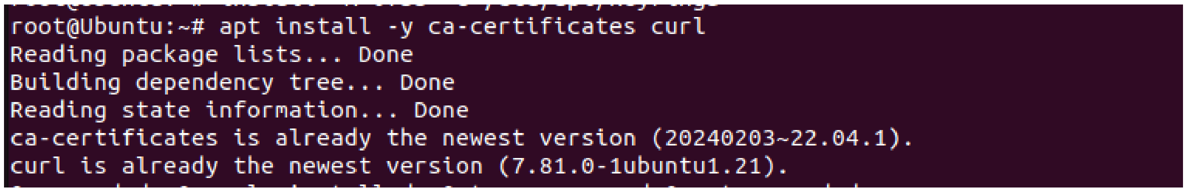
```
root@Ubuntu:~# apt update
Hit:1 http://co.archive.ubuntu.com/ubuntu jammy InRelease
Hit:2 http://co.archive.ubuntu.com/ubuntu jammy-updates InRelease
Hit:3 http://co.archive.ubuntu.com/ubuntu jammy-backports InRelease
Get:4 http://security.ubuntu.com/ubuntu jammy-security InRelease [129 kB]
Get:5 http://security.ubuntu.com/ubuntu jammy-security/main amd64 c-n-f Metadata [13,9 kB]
Get:6 http://security.ubuntu.com/ubuntu jammy-security/universe amd64 c-n-f Metadata [22,3 kB]
Fetched 165 kB in 2s (66,3 kB/s)
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
All packages are up to date.
```



```
root@Ubuntu:~# apt upgrade -y
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
Calculating upgrade... Done
0 upgraded, 0 newly installed, 0 to remove and 0 not upgraded.
```

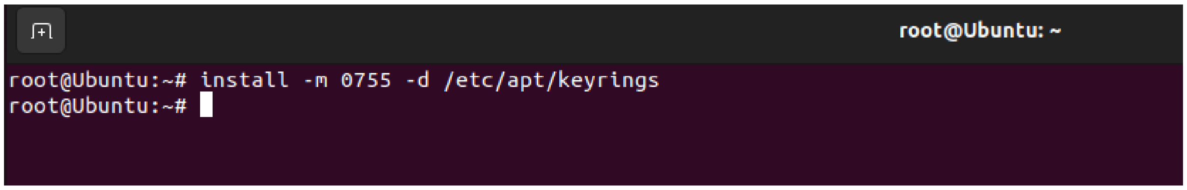
2. **Instalar Docker Engine (Repositorio)**

2.1 Paquetes necesarios para añadir repositorios seguros y descargar llaves.
(instrucciones recomendadas por el Docker oficial)
apt install -y ca-certificates curl



```
root@Ubuntu:~# apt install -y ca-certificates curl
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
ca-certificates is already the newest version (20240203~22.04.1).
curl is already the newest version (7.81.0-1ubuntu1.21).
```

2.2 Crear el directorio /etc/apt/keyrings con permisos apropiados, ahí se guardará la llave GPG en formato seguro.
Que es una llave GPG (GNU Privacy Guard) es una **clave criptográfica** que se usa para verificar la autenticidad e integridad de archivos o mensajes. En el contexto de Linux y APT, se usa principalmente para **verificar que los paquetes que descargas del repositorio realmente fueron creados por el proveedor oficial y no fueron modificados**.
install -m 0755 -d /etc/apt/keyrings

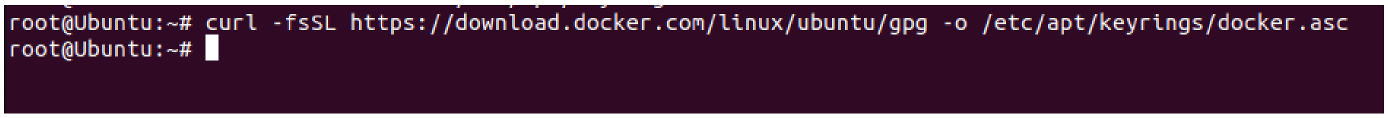


root@Ubuntu: ~

```
root@Ubuntu:~# install -m 0755 -d /etc/apt/keyrings
root@Ubuntu:~# 
```

2.3 Descargar la llave GPG oficial de Docker y se guarda en la ubicación anteriormente creada

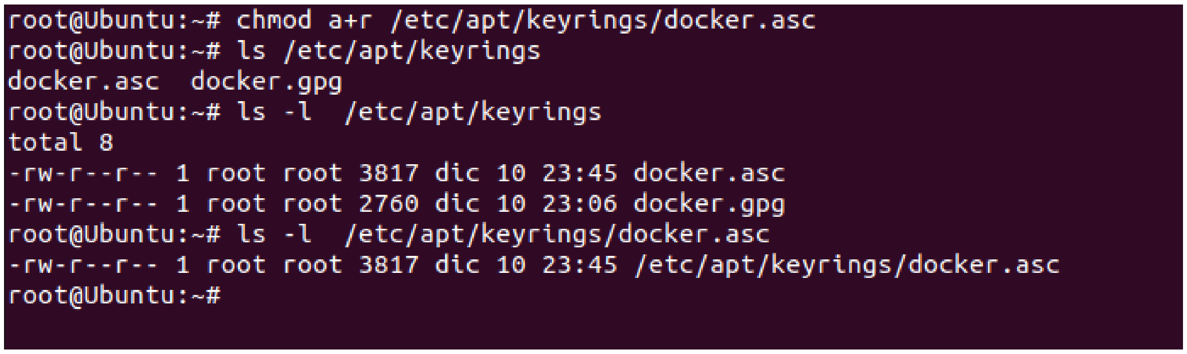
curl -fsSL https://download.docker.com/linux/ubuntu/gpg -o /etc/apt/keyrings/docker.asc



```
root@Ubuntu:~# curl -fsSL https://download.docker.com/linux/ubuntu/gpg -o /etc/apt/keyrings/docker.asc
root@Ubuntu:~# 
```

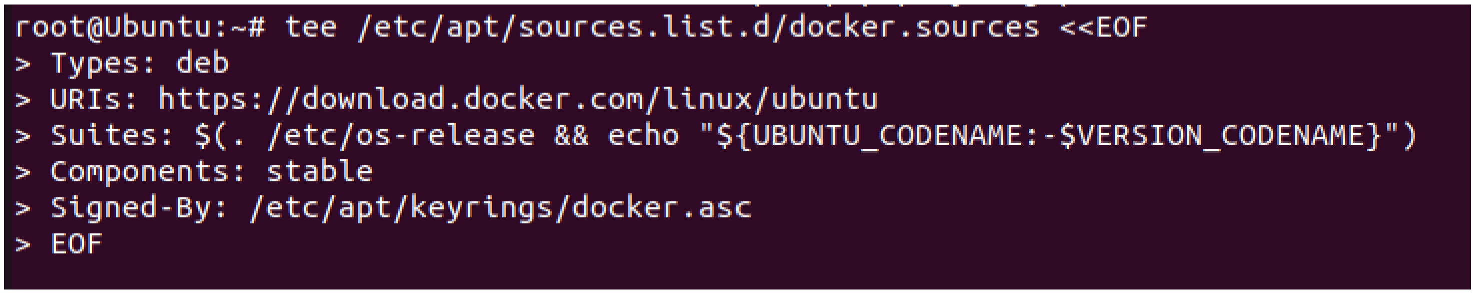
2.4 Hay que asegurar que el archive sea legible por APT lo cual evita errores de permisos

chmod a+r /etc/apt/keyrings/docker.asc



```
root@Ubuntu:~# chmod a+r /etc/apt/keyrings/docker.asc
root@Ubuntu:~# ls /etc/apt/keyrings
docker.asc  docker.gpg
root@Ubuntu:~# ls -l  /etc/apt/keyrings
total 8
-rw-r--r-- 1 root root 3817 dic 10 23:45 docker.asc
-rw-r--r-- 1 root root 2760 dic 10 23:06 docker.gpg
root@Ubuntu:~# ls -l  /etc/apt/keyrings/docker.asc
-rw-r--r-- 1 root root 3817 dic 10 23:45 /etc/apt/keyrings/docker.asc
root@Ubuntu:~#
```

2.5 Crear un archive de repositorio para apt con el formato sources.list.d



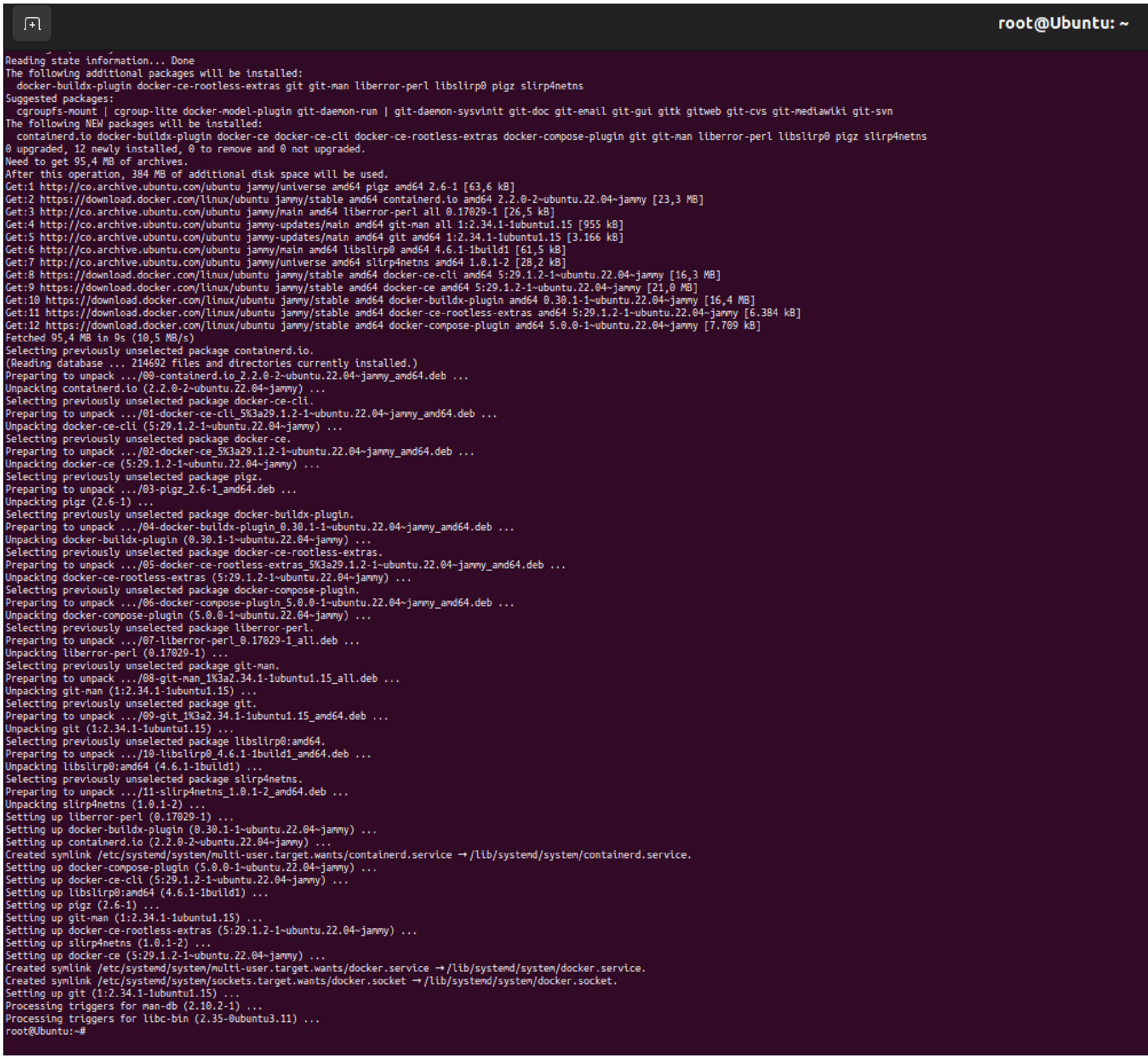
```
root@Ubuntu:~# tee /etc/apt/sources.list.d/docker.sources <<EOF
> Types: deb
> URIs: https://download.docker.com/linux/ubuntu
> Suites: $(. /etc/os-release && echo "${UBUNTU_CODENAME:-$VERSION_CODENAME}")
> Components: stable
> Signed-By: /etc/apt/keyrings/docker.asc
> EOF
```

Y por ultimo actualizamos con apt update para actualizar la lista de paquetes con el nuevo repositorio de docker

3. **Instalación de Docker Engine**

3.1 Instalamos el Docker engine y sus componentes

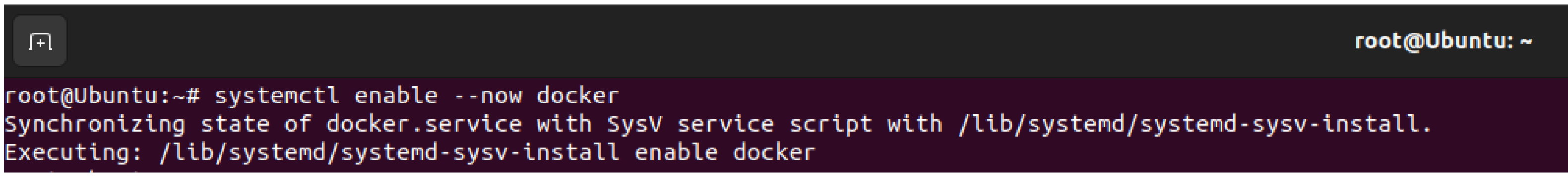
apt install -y docker-ce docker-ce-cli containerd.io docker-compose-plugin



```
Reading state information... Done
The following additional packages will be installed:
  docker-buildx-plugin docker-ce-rootless-extras git git-man liberror-perl libslirp0 pigz slirp4netns
Suggested packages:
  cgroupfs-mount | cgroup-lite docker-model-plugin git-daemon-run | git-daemon-sysvinit git-doc git-email git-gui gitk gitweb git-cvs git-mediawiki git-svn
The following NEW packages will be installed:
  containerd.io docker-buildx-plugin docker-ce docker-ce-cli docker-ce-rootless-extras docker-compose-plugin git git-man liberror-perl libslirp0 pigz slirp4netns
0 upgraded, 12 newly installed, 0 to remove and 0 not upgraded.
Need to get 95,4 MB of archives.
After this operation, 384 MB of additional disk space will be used.
Get:1 http://co.archive.ubuntu.com/ubuntu jammy/universe amd64 pigz amd64 2.6-1 [63,6 kB]
Get:2 https://download.docker.com/linux/ubuntu jammy/stable amd64 containerd.io amd64 2.2.0-2~ubuntu.22.04~jammy [23,3 MB]
Get:3 http://co.archive.ubuntu.com/ubuntu jammy/main amd64 liberror-perl all 0.17029-1 [26,5 kB]
Get:4 http://co.archive.ubuntu.com/ubuntu jammy-updates/main amd64 git-man all 1:2.34.1-1ubuntu1.15 [955 kB]
Get:5 http://co.archive.ubuntu.com/ubuntu jammy-updates/main amd64 git amd64 1:2.34.1-1ubuntu1.15 [3.166 kB]
Get:6 http://co.archive.ubuntu.com/ubuntu jammy/main amd64 libslirp0 amd64 4.6.1-1build1 [61,5 kB]
Get:7 http://co.archive.ubuntu.com/ubuntu jammy/universe amd64 slirp4netns amd64 1.0.1-2 [28,2 kB]
Get:8 https://download.docker.com/linux/ubuntu jammy/stable amd64 docker-ce-cli amd64 5:29.1.2-1~ubuntu.22.04~jammy [16,3 MB]
Get:9 https://download.docker.com/linux/ubuntu jammy/stable amd64 docker-ce amd64 5:29.1.2-1~ubuntu.22.04~jammy [21,0 MB]
Get:10 https://download.docker.com/linux/ubuntu jammy/stable amd64 docker-buildx-plugin amd64 0.30.1-1~ubuntu.22.04~jammy [16,4 MB]
Get:11 https://download.docker.com/linux/ubuntu jammy/stable amd64 docker-ce-rootless-extras amd64 5:29.1.2-1~ubuntu.22.04~jammy [6.384 kB]
Get:12 https://download.docker.com/linux/ubuntu jammy/stable amd64 docker-compose-plugin amd64 5.0.0-1~ubuntu.22.04~jammy [7.709 kB]
Fetched 95,4 MB in 9s (10,5 MB/s)
Selecting previously unselected package containerd.io.
(Reading database ... 214692 files and directories currently installed.)
Preparing to unpack .../00-containerd.io_2.2.0-2~ubuntu.22.04~jammy_amd64.deb ...
Unpacking containerd.io (2.2.0-2~ubuntu.22.04~jammy) ...
Selecting previously unselected package docker-ce-cli.
Preparing to unpack .../01-docker-ce-cli_5%3a29.1.2-1~ubuntu.22.04~jammy_amd64.deb ...
Unpacking docker-ce-cli (5:29.1.2-1~ubuntu.22.04~jammy) ...
Selecting previously unselected package docker-ce.
Preparing to unpack .../02-docker-ce_5%3a29.1.2-1~ubuntu.22.04~jammy_amd64.deb ...
Unpacking docker-ce (5:29.1.2-1~ubuntu.22.04~jammy) ...
Selecting previously unselected package pigz.
Preparing to unpack .../03-pigz_2.6-1_amd64.deb ...
Unpacking pigz (2.6-1) ...
Selecting previously unselected package docker-buildx-plugin.
Preparing to unpack .../04-docker-buildx-plugin_0.30.1-1~ubuntu.22.04~jammy_amd64.deb ...
Unpacking docker-buildx-plugin (0.30.1-1~ubuntu.22.04~jammy) ...
Selecting previously unselected package docker-ce-rootless-extras.
Preparing to unpack .../05-docker-ce-rootless-extras_5%3a29.1.2-1~ubuntu.22.04~jammy_amd64.deb ...
Unpacking docker-ce-rootless-extras (5:29.1.2-1~ubuntu.22.04~jammy) ...
Selecting previously unselected package docker-compose-plugin.
Preparing to unpack .../06-docker-compose-plugin_5.0.0-1~ubuntu.22.04~jammy_amd64.deb ...
Unpacking docker-compose-plugin (5.0.0-1~ubuntu.22.04~jammy) ...
Selecting previously unselected package liberror-perl.
Preparing to unpack .../07-liberror-perl_0.17029-1_all.deb ...
Unpacking liberror-perl (0.17029-1) ...
Selecting previously unselected package git-man.
Preparing to unpack .../08-git-man_1%3a2.34.1-1ubuntu1.15_all.deb ...
Unpacking git-man (1:2.34.1-1ubuntu1.15) ...
Selecting previously unselected package git.
Preparing to unpack .../09-git_1%3a2.34.1-1ubuntu1.15_amd64.deb ...
Unpacking git (1:2.34.1-1ubuntu1.15) ...
Selecting previously unselected package libslirp0:amd64.
Preparing to unpack .../10-libslirp0_4.6.1-1build1_amd64.deb ...
Unpacking libslirp0:amd64 (4.6.1-1build1) ...
Selecting previously unselected package slirp4netns.
Preparing to unpack .../11-slirp4netns_1.0.1-2_amd64.deb ...
Unpacking slirp4netns (1.0.1-2) ...
Setting up liberror-perl (0.17029-1) ...
Setting up docker-buildx-plugin (0.30.1-1~ubuntu.22.04~jammy) ...
Setting up containerd.io (2.2.0-2~ubuntu.22.04~jammy) ...
Created symlink /etc/systemd/system/multi-user.target.wants/containerd.service → /lib/systemd/system/containerd.service.
Setting up docker-compose-plugin (5.0.0-1~ubuntu.22.04~jammy) ...
Setting up docker-ce-cli (5:29.1.2-1~ubuntu.22.04~jammy) ...
Setting up libslirp0:amd64 (4.6.1-1build1) ...
Setting up pigz (2.6-1) ...
Setting up git-man (1:2.34.1-1ubuntu1.15) ...
Setting up docker-ce-rootless-extras (5:29.1.2-1~ubuntu.22.04~jammy) ...
Setting up slirp4netns (1.0.1-2) ...
Setting up docker-ce (5:29.1.2-1~ubuntu.22.04~jammy) ...
Created symlink /etc/systemd/system/multi-user.target.wants/docker.service → /lib/systemd/system/docker.service.
Created symlink /etc/systemd/system/sockets.target.wants/docker.socket → /lib/systemd/system/docker.socket.
Setting up git (1:2.34.1-1ubuntu1.15) ...
Processing triggers for man-db (2.10.2-1) ...
Processing triggers for libc-bin (2.35-0ubuntu3.11) ...
root@Ubuntu:~#
```

3.2 activar el servicio para que arranque en boot

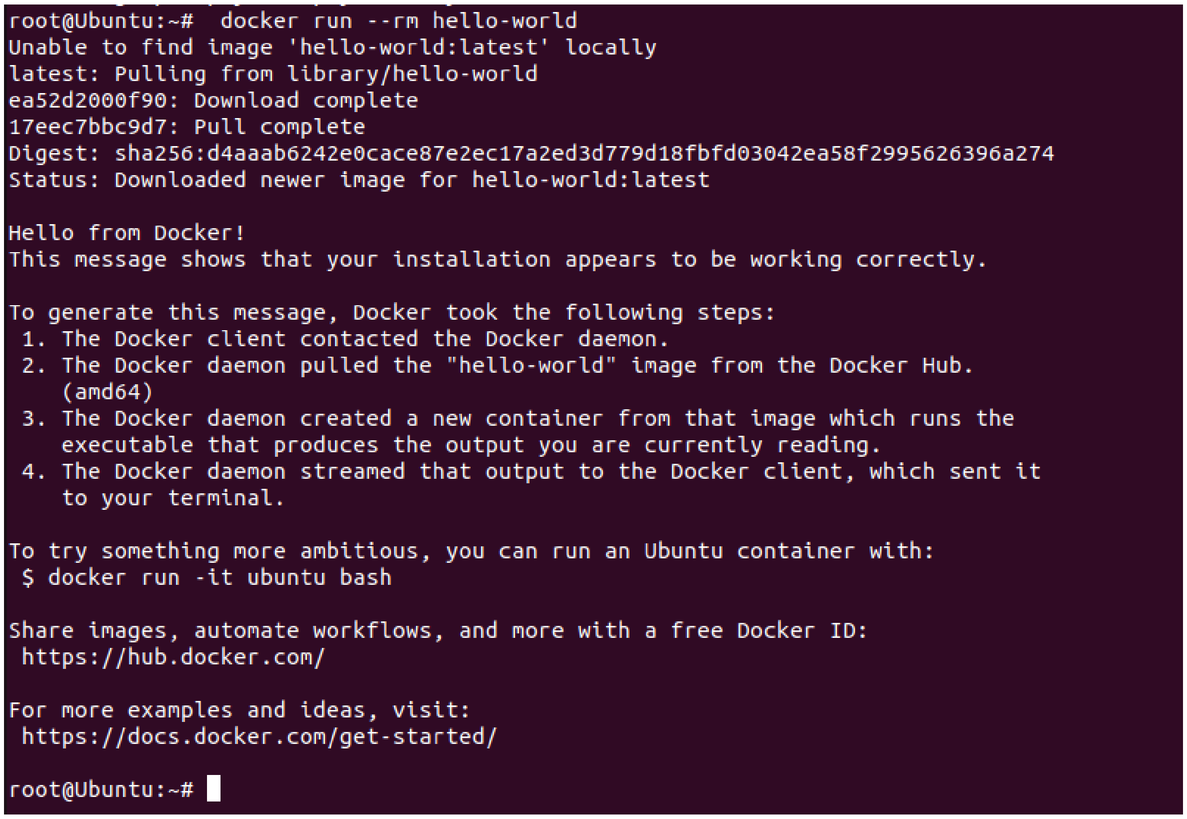
systemctl enable --now Docker



```
root@Ubuntu:~# systemctl enable --now docker
Synchronizing state of docker.service with SysV service script with /lib/systemd/systemd-sysv-install.
Executing: /lib/systemd/systemd-sysv-install enable docker
```

3.3 Prueba practica (ejecutar contenedor “hello world”):

docker run --rm hello-world

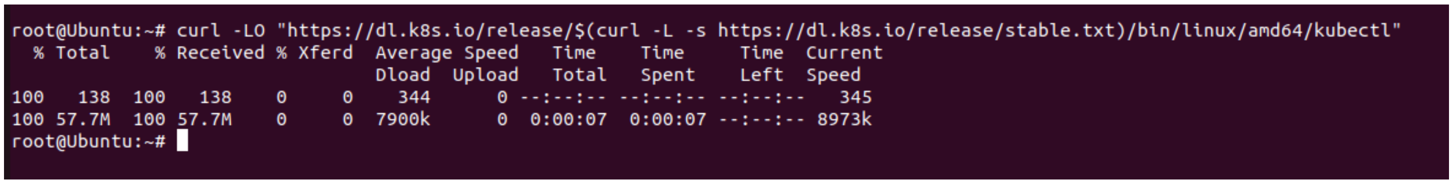


```
root@Ubuntu:~#  docker run --rm hello-world
Unable to find image 'hello-world:latest' locally
latest: Pulling from library/hello-world
ea52d2000f90: Download complete
17eec7bbc9d7: Pull complete
Digest: sha256:d4aaab6242e0cace87e2ec17a2ed3d779d18fbfd03042ea58f2995626396a274
Status: Downloaded newer image for hello-world:latest

Hello from Docker!
This message shows that your installation appears to be working correctly.

To generate this message, Docker took the following steps:
 1. The Docker client contacted the Docker daemon.
 2. The Docker daemon pulled the "hello-world" image from the Docker Hub.
    (amd64)
 3. The Docker daemon created a new container from that image which runs the
    executable that produces the output you are currently reading.
 4. The Docker daemon streamed that output to the Docker client, which sent it
    to your terminal.

To try something more ambitious, you can run an Ubuntu container with:
 $ docker run -it ubuntu bash

Share images, automate workflows, and more with a free Docker ID:
 https://hub.docker.com/

For more examples and ideas, visit:
 https://docs.docker.com/get-started/

root@Ubuntu:~#
```

4. **Instalar kubectl**

4.1 Instalación del binario kubectl (Descarga la versión estable más reciente)

curl -LO https://dl.k8s.io/release/$(curl -L -s https://dl.k8s.io/release/stable.txt)/bin/linux/amd64/kubectl

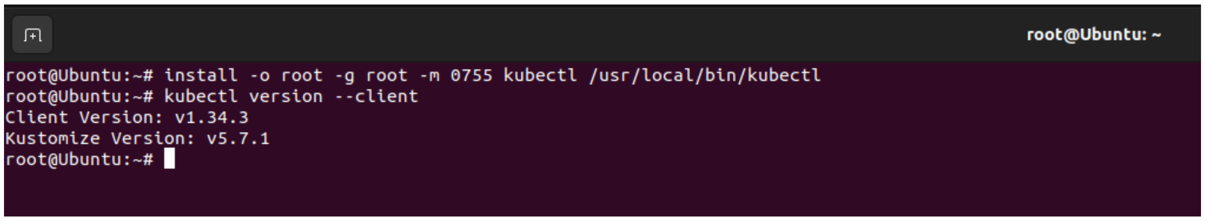


```
root@Ubuntu:~# curl -LO "https://dl.k8s.io/release/$(curl -L -s https://dl.k8s.io/release/stable.txt)/bin/linux/amd64/kubectl"
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100   138  100   138    0     0    344      0 --:--:-- --:--:-- --:--:--   345
100 57.7M  100 57.7M    0     0  7900k      0  0:00:07  0:00:07 --:--:-- 8973k
root@Ubuntu:~#
```

4.2 copiar e instalar kubectl a /usr/local/bin y se le aplica permisos ejecutables y propiedad para root

install -o root -g root -m 0755 kubectl /usr/local/bin/kubectl

y comprobamos la versión **kubectl version --client**

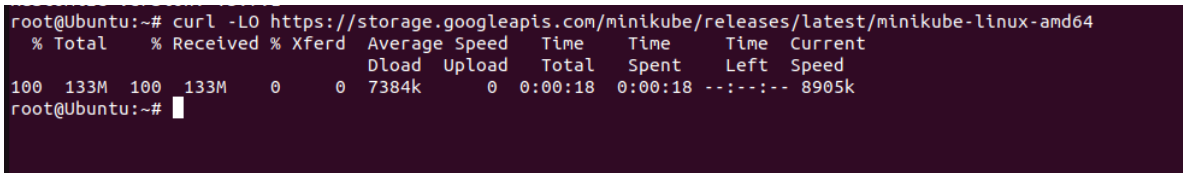


```
root@Ubuntu: ~
root@Ubuntu:~# install -o root -g root -m 0755 kubectl /usr/local/bin/kubectl
root@Ubuntu:~# kubectl version --client
Client Version: v1.34.3
Kustomize Version: v5.7.1
root@Ubuntu:~#
```

5. **Instalar Minikube**

   5.1 Descargar la última versión de minikube

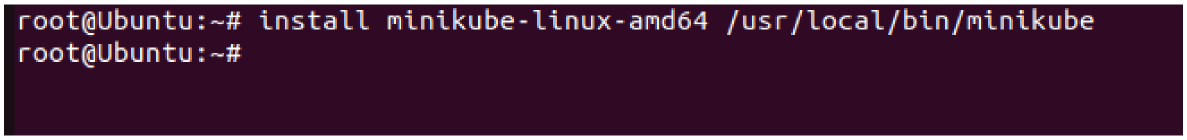
   curl -LO https://storage.googleapis.com/minikube/releases/latest/minikube-linux-amd64



```
root@Ubuntu:~# curl -LO https://storage.googleapis.com/minikube/releases/latest/minikube-linux-amd64
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100  133M  100  133M    0     0  7384k      0  0:00:18  0:00:18 --:--:-- 8905k
root@Ubuntu:~# 
```

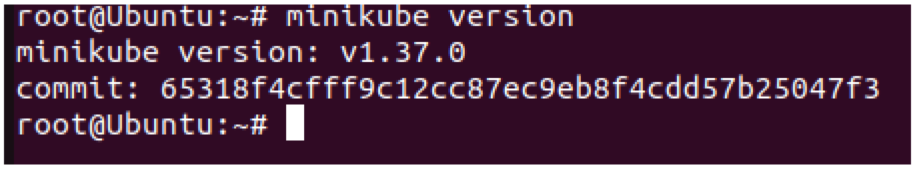
   5.2 Instalar minikube en /usr/local/bin y lo hace ejecutable

   install minikube-linux-amd64 /usr/local/bin/minikube



```
root@Ubuntu:~# install minikube-linux-amd64 /usr/local/bin/minikube
root@Ubuntu:~#
```

   5.3 Verificar la version minikube con **minikube version**

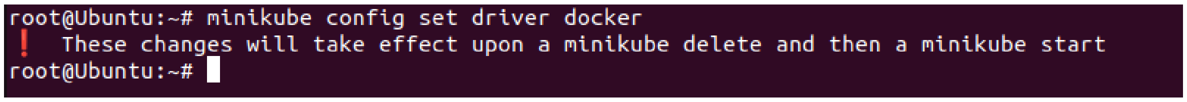


```
root@Ubuntu:~# minikube version
minikube version: v1.37.0
commit: 65318f4cfff9c12cc87ec9eb8f4cdd57b25047f3
root@Ubuntu:~# 
```

6. **Configurar driver por defecto a Docker**

   Fijamos a Docker como driver por defecto con

   minikube config set driver Docker



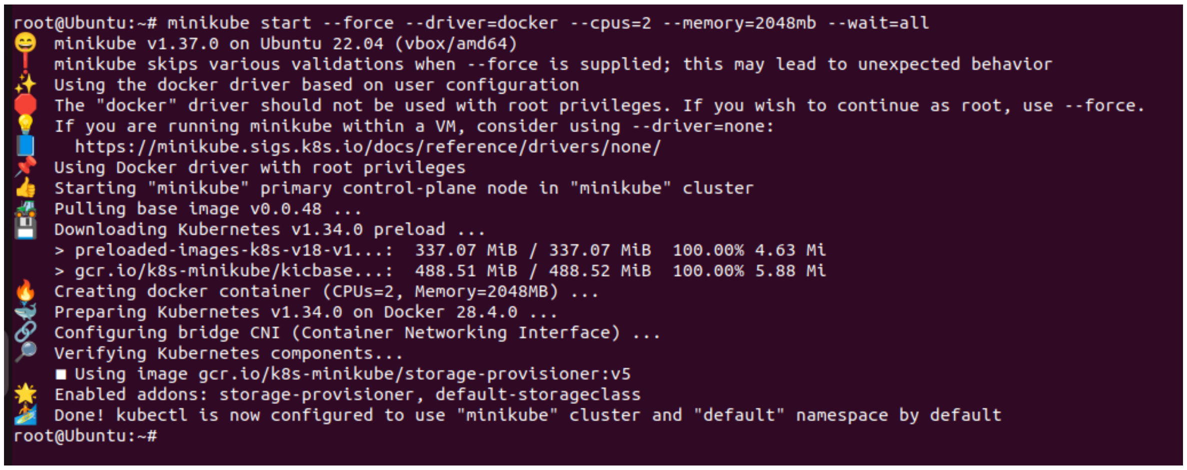
```
root@Ubuntu:~# minikube config set driver docker
❗  These changes will take effect upon a minikube delete and then a minikube start
root@Ubuntu:~# 
```

   La traducción del mensaje es: “El cambio de driver solo se aplicará cuando borres el cluster actual (si existe) y luego arranques Minikube nuevamente.” Si no se ha iniciado minikube no tendríamos que usar el **minikube delete** y volverlo a iniciar con **minikube start –driver=Docker,** solo se inicia

7. **Iniciar cluster: minikube start**

   7.1 Iniciamos el cluster. Ajustando el **– -cpus=2 - -memory=2048mb - -wait=all**

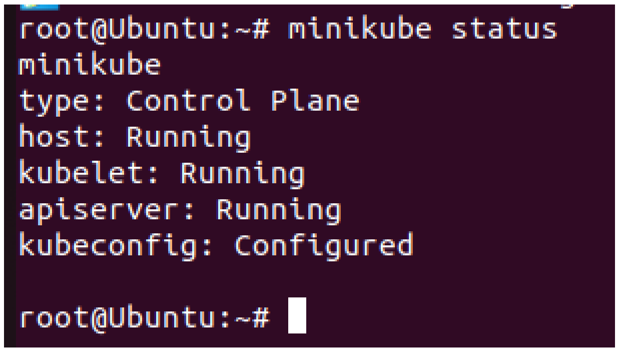
   minikube start --driver=docker --cpus=2 --memory=2048mb --wait=all



```
root@Ubuntu:~# minikube start --force --driver=docker --cpus=2 --memory=2048mb --wait=all
    minikube v1.37.0 on Ubuntu 22.04 (vbox/amd64)
    minikube skips various validations when --force is supplied; this may lead to unexpected behavior
    Using the docker driver based on user configuration
    The "docker" driver should not be used with root privileges. If you wish to continue as root, use --force.
    If you are running minikube within a VM, consider using --driver=none:
      https://minikube.sigs.k8s.io/docs/reference/drivers/none/
    Using Docker driver with root privileges
    Starting "minikube" primary control-plane node in "minikube" cluster
    Pulling base image v0.0.48 ...
    Downloading Kubernetes v1.34.0 preload ...
    > preloaded-images-k8s-v18-v1...:  337.07 MiB / 337.07 MiB  100.00% 4.63 Mi
    > gcr.io/k8s-minikube/kicbase...:  488.51 MiB / 488.52 MiB  100.00% 5.88 Mi
    Creating docker container (CPUs=2, Memory=2048MB) ...
    Preparing Kubernetes v1.34.0 on Docker 28.4.0 ...
    Configuring bridge CNI (Container Networking Interface) ...
    Verifying Kubernetes components...
    ■ Using image gcr.io/k8s-minikube/storage-provisioner:v5
    Enabled addons: storage-provisioner, default-storageclass
    Done! kubectl is now configured to use "minikube" cluster and "default" namespace by default
root@Ubuntu:~#
```

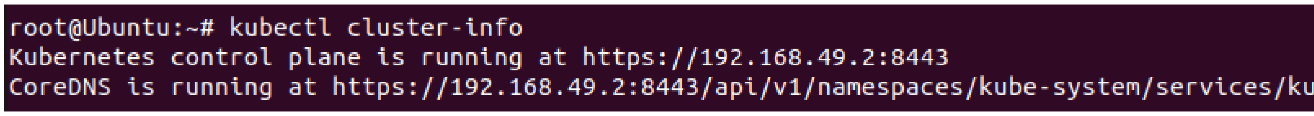
7.2 Comprobamos el estado de Minikube:

Minikube status



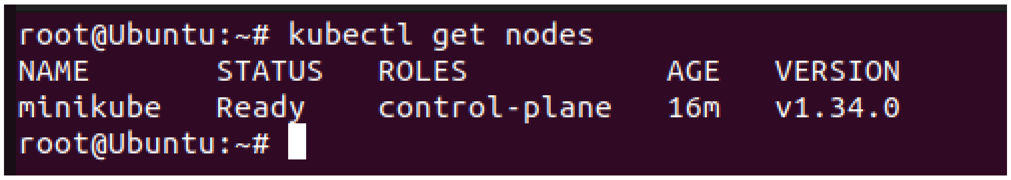
```
root@Ubuntu:~# minikube status
minikube
type: Control Plane
host: Running
kubelet: Running
apiserver: Running
kubeconfig: Configured

root@Ubuntu:~#
```

7.3 Verificar con **kubctl cluster-info**



```
root@Ubuntu:~# kubectl cluster-info
Kubernetes control plane is running at https://192.168.49.2:8443
CoreDNS is running at https://192.168.49.2:8443/api/v1/namespaces/kube-system/services/ku
```

7.4 Listado de nodos con **kubectl get nodes**



```
root@Ubuntu:~# kubectl get nodes
NAME       STATUS   ROLES           AGE   VERSION
minikube   Ready    control-plane   16m   v1.34.0
root@Ubuntu:~#
```