



IES Ribera del Tajo

ADMINISTRACIÓN DE SISTEMAS INFORMÁTICOS EN RED

FLOW RIBERA

Autor:
Equipo Azul

2022/23

ÍNDICE

1. Introducción	2
2. Interfaz web	3
3. Servidor	4
3.1. Apache	5
3.2. MariaDB	8
4. Cluster	15

Introducción

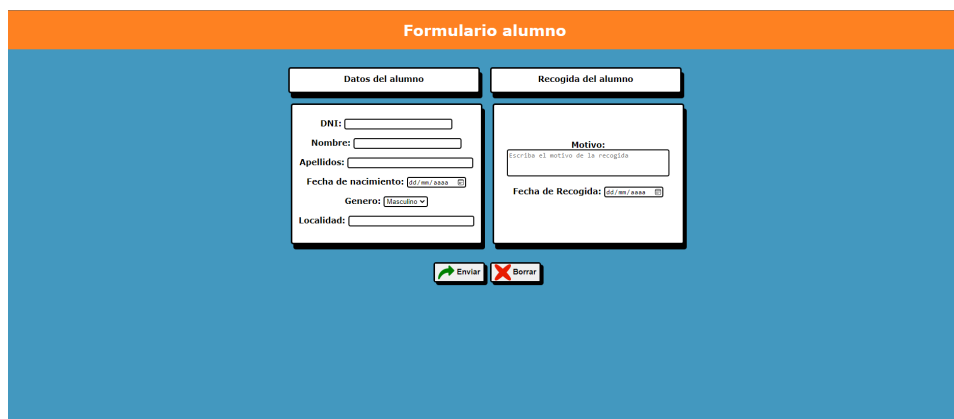
Nuestro equipo ha sido contactado por los responsables del *IES Ribera del Tajo* para ayudarles con la gestión de las salidas de los alumnos del recinto.

Nos han pedido que les facilitemos el modo en el que llevan el registro de los alumnos que salen del centro, y para ello vamos a ofrecerles una aplicación web alojada en el servidor local del centro cuyo objetivo es automatizar lo máximo posible el registro.

A lo largo de este documento vamos a exponer el proceso por el cual hemos pasado para configurar todo el sistema, tanto la parte del servidor y de la base de datos, como de la interfaz de la aplicación; y el funcionamiento de este.

Interfaz web

Nuestro objetivo a la hora de realizar la interfaz desde la que se va a realizar el registro, es que sea lo más simple posible para así ayudar a los responsables y complicarles lo menos posible.



The image shows a web form titled "Formulario alumno" with an orange header. The form is divided into two main sections: "Datos del alumno" and "Recogida del alumno".

Datos del alumno:

- DNI:
- Nombre:
- Apellidos:
- Fecha de nacimiento:
- Genero:
- Localidad:

Recogida del alumno:

- Motivo:
- Fecha de Recogida:

At the bottom of the form, there are two buttons: "Enviar" (with a green arrow icon) and "Borrar" (with a red X icon).

Figura 2.1: Interfaz

Servidor

El servidor en el que va a estar alojada la aplicación consistirá en un cluster formado por 2 nodos con un sistema operativo *Linux*.

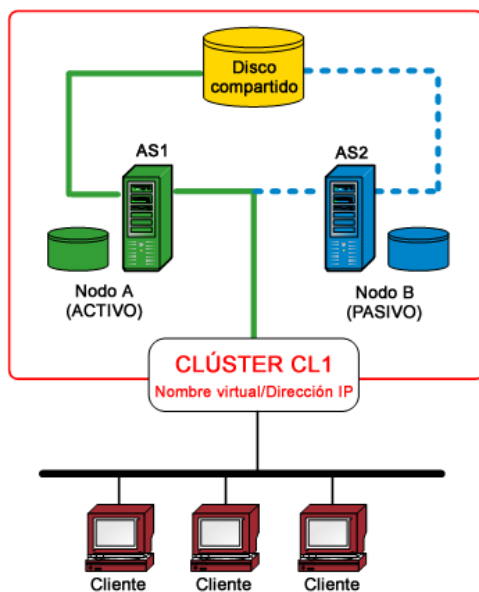


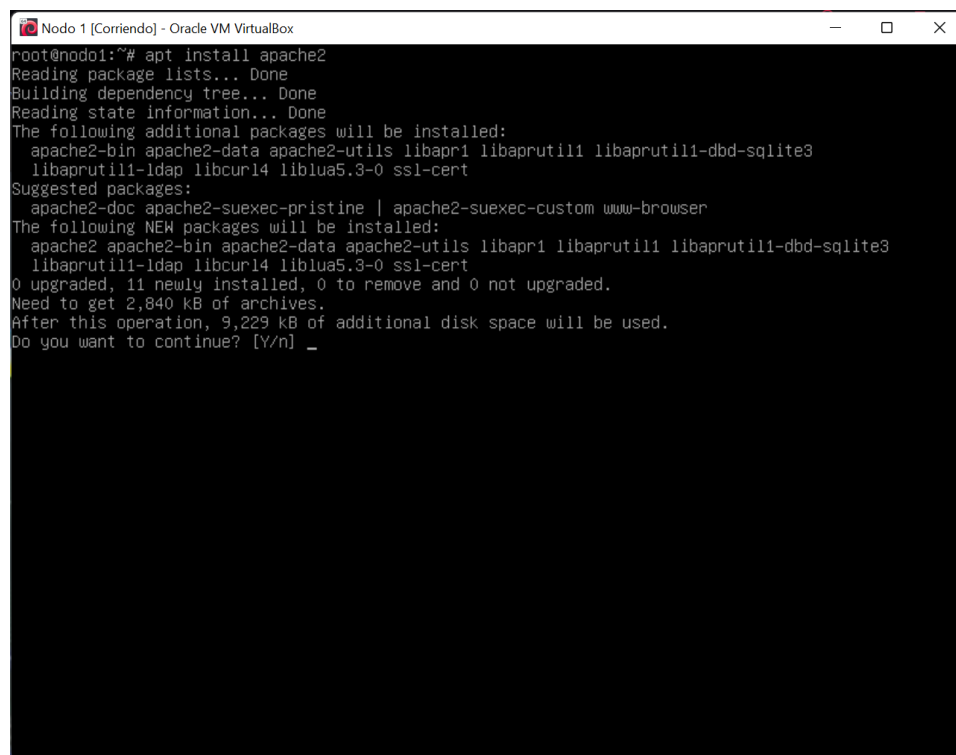
Figura 3.1: Arquitectura de cluster

En el cluster, instalaremos *Apache* para acceder a la interfaz web y una base de datos, en este caso *MariaDB* para almacenar los datos requeridos.

3.1. Apache

Para instalar *Apache*, lo único que debemos hacer es realizar el siguiente comando:

```
apt install apache2
```



```
Nodo 1 [Corriendo] - Oracle VM VirtualBox
root@nodo1:~# apt install apache2
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  apache2-bin apache2-data apache2-utils libapr1 libaprutil1 libaprutil1-dbd-sqlite3
  libaprutil1-ldap libcurl4 liblua5.3-0 ssl-cert
Suggested packages:
  apache2-doc apache2-suexec-pristine | apache2-suexec-custom www-browser
The following NEW packages will be installed:
  apache2 apache2-bin apache2-data apache2-utils libapr1 libaprutil1 libaprutil1-dbd-sqlite3
  libaprutil1-ldap libcurl4 liblua5.3-0 ssl-cert
0 upgraded, 11 newly installed, 0 to remove and 0 not upgraded.
Need to get 2,840 kB of archives.
After this operation, 9,229 kB of additional disk space will be used.
Do you want to continue? [Y/n] _
```

Figura 3.2: Instalación de Apache

Para comprobar que *Apache* se ha instalado correctamente, accedemos al servidor a través del navegador.

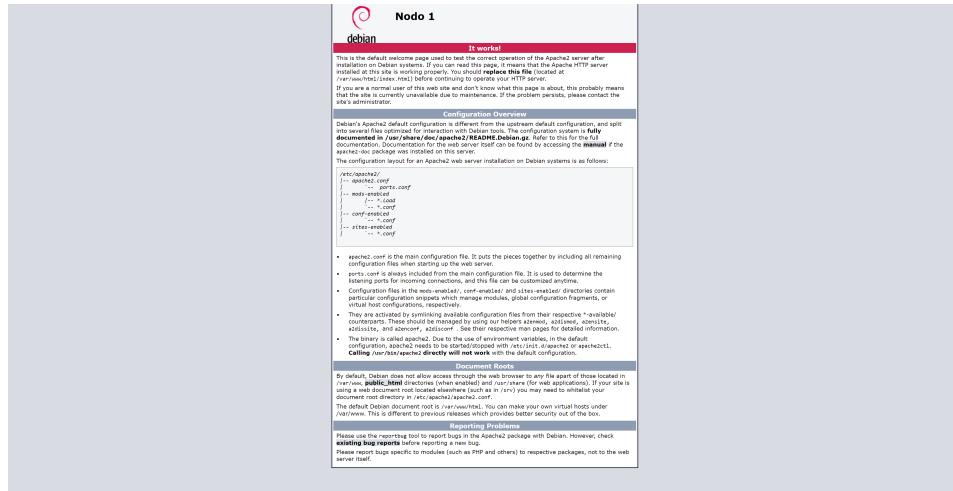


Figura 3.3: Web nodo 1

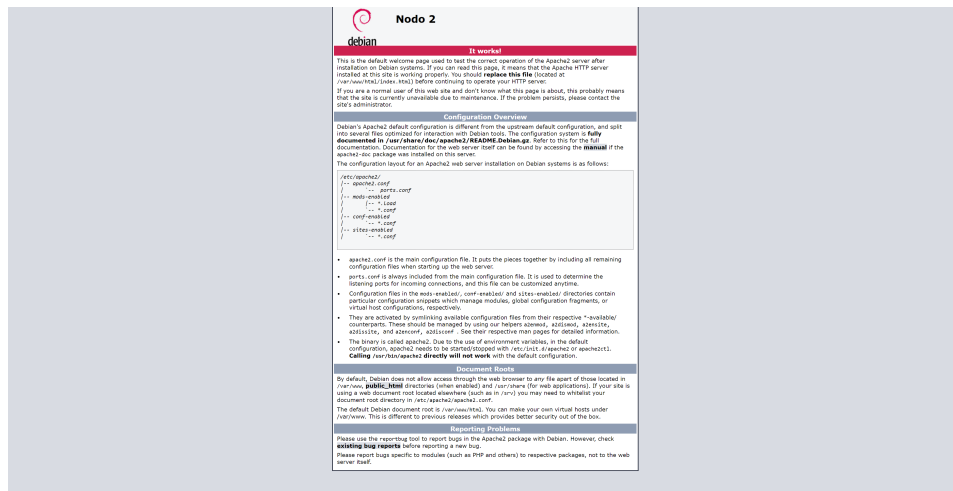


Figura 3.4: Web nodo 2

Una vez que hemos comprobado que los servidores web funcionan correctamente, procedemos a modificar la web en cada uno de los servidores.



The image shows a web form titled "Formulario alumno" with an orange header. The form is divided into two main sections: "Datos del alumno" and "Recogida del alumno".

Datos del alumno:

- DNI:
- Nombre:
- Apellidos:
- Fecha de nacimiento:
- Genero:
- Localidad:

Recogida del alumno:

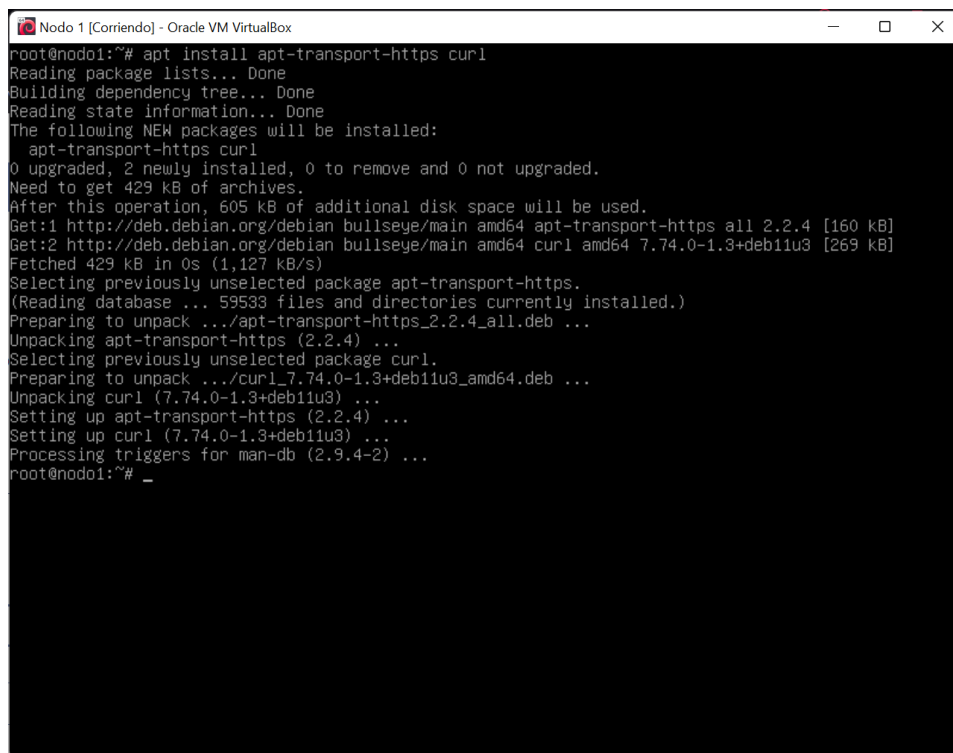
- Motivo:
- Fecha de Recogida:

At the bottom of the form, there are two buttons: "Enviar" (with a green arrow icon) and "Borrar" (with a red X icon).

Figura 3.5: Interfaz modificada

3.2. MariaDB

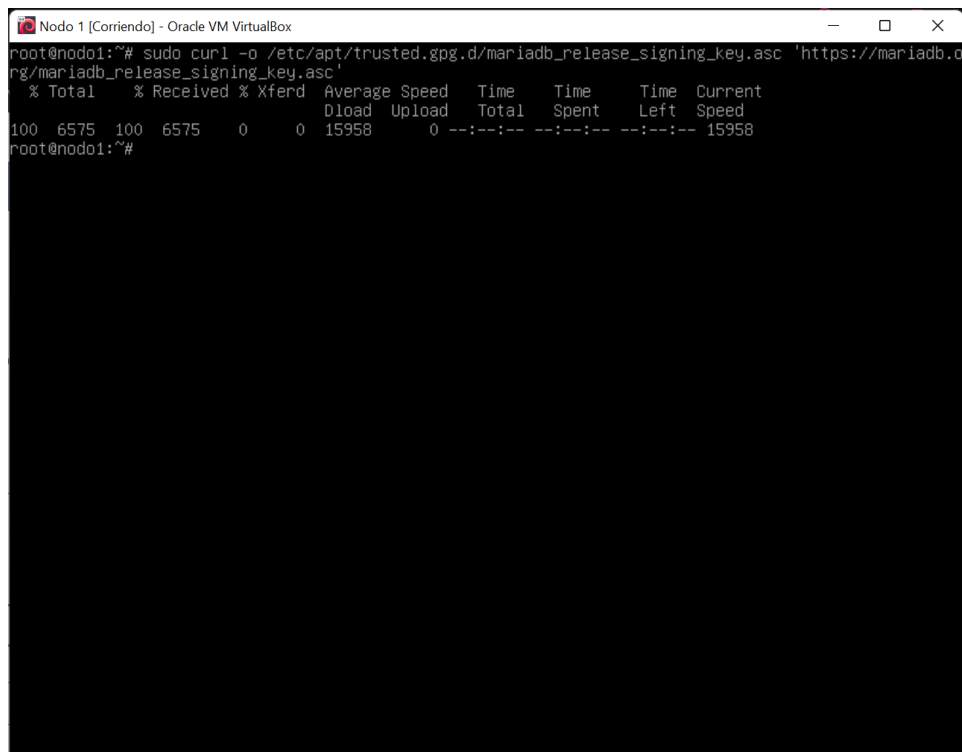
Antes de comenzar con la instalación de *MariaDB*, necesitamos instalar otras herramientas. Las que vamos a instalar son *apt-transport-https* y *curl* para poder obtener las claves necesarias para la instalación.



```
Nodo 1 [Corriendo] - Oracle VM VirtualBox
root@nodo1:~# apt install apt-transport-https curl
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following NEW packages will be installed:
  apt-transport-https curl
0 upgraded, 2 newly installed, 0 to remove and 0 not upgraded.
Need to get 429 kB of archives.
After this operation, 605 kB of additional disk space will be used.
Get:1 http://deb.debian.org/debian bullseye/main amd64 apt-transport-https all 2.2.4 [160 kB]
Get:2 http://deb.debian.org/debian bullseye/main amd64 curl amd64 7.74.0-1.3+deb11u3 [269 kB]
Fetched 429 kB in 0s (1,127 kB/s)
Selecting previously unselected package apt-transport-https.
(Reading database ... 59533 files and directories currently installed.)
Preparing to unpack .../apt-transport-https_2.2.4_all.deb ...
Unpacking apt-transport-https (2.2.4) ...
Selecting previously unselected package curl.
Preparing to unpack .../curl_7.74.0-1.3+deb11u3_amd64.deb ...
Unpacking curl (7.74.0-1.3+deb11u3) ...
Setting up apt-transport-https (2.2.4) ...
Setting up curl (7.74.0-1.3+deb11u3) ...
Processing triggers for man-db (2.9.4-2) ...
root@nodo1:~# _
```

Figura 3.6: Instalación de *apt-transport-https* y *curl*

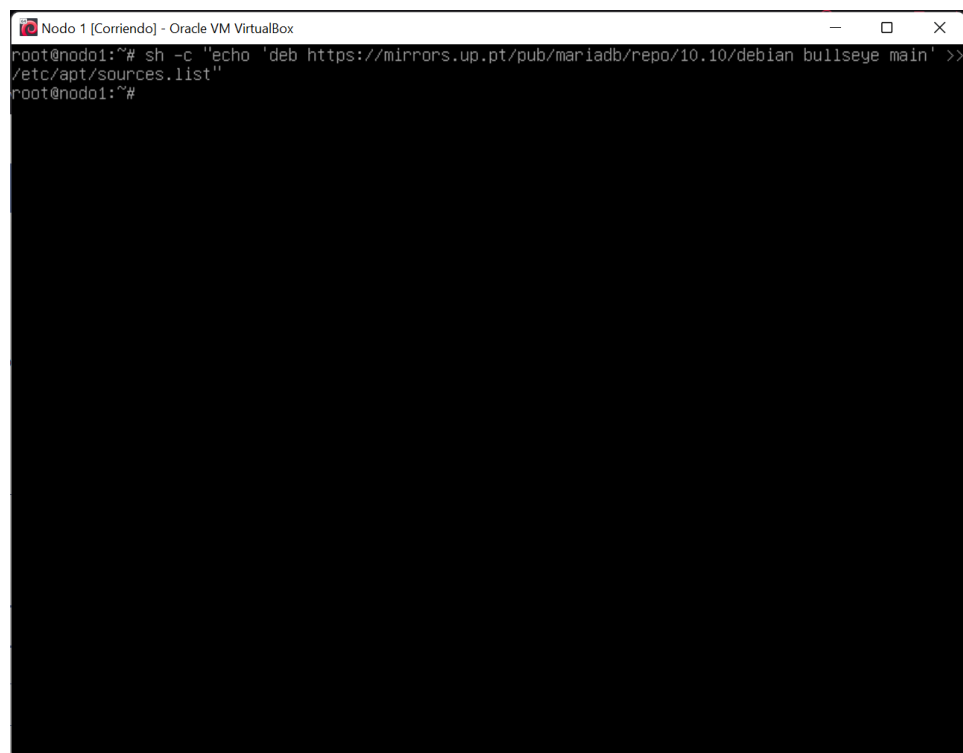
Una vez que tenemos las herramientas instaladas, obtenemos las claves necesarias para la instalación.



```
Nodo 1 [Corriendo] - Oracle VM VirtualBox
root@nodo1:~# sudo curl -o /etc/apt/trusted.gpg.d/mariadb_release_signing_key.asc 'https://mariadb.org/mariadb_release_signing_key.asc'
% Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
           Dload  Upload   Total   Spent    Left   Speed
100 6575  100 6575    0     0  15958      0 --:--:-- --:--:-- --:--:-- 15958
root@nodo1:~#
```

Figura 3.7: Obtención de las claves

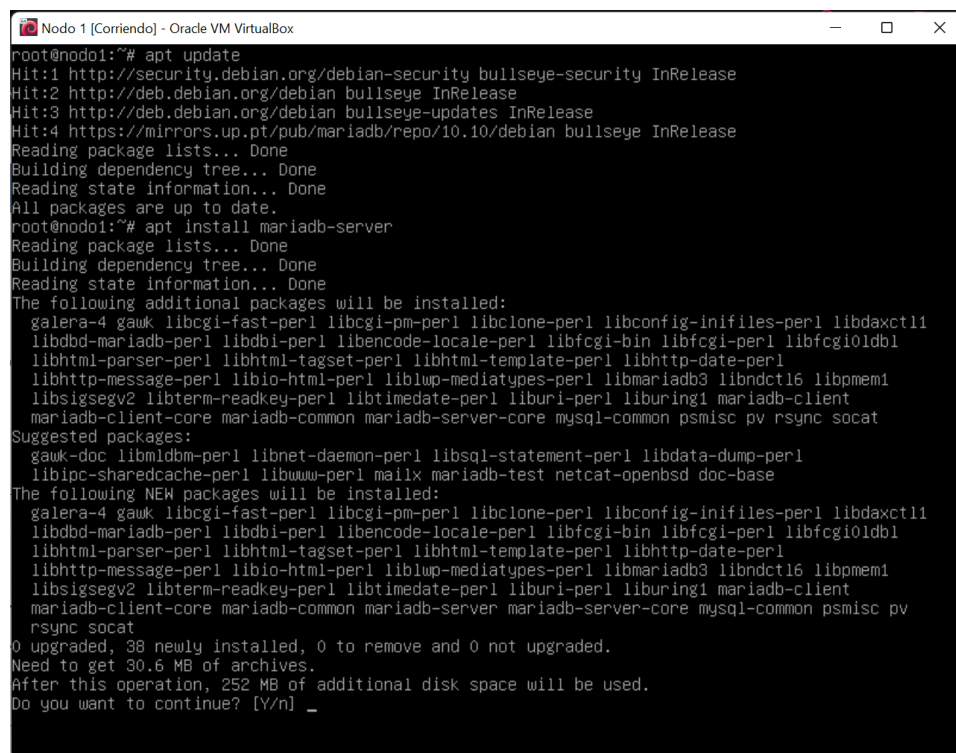
Después de obtener las claves, agregamos los repositorios de *MariaDB*.



```
Nodo 1 [Corriendo] - Oracle VM VirtualBox
root@nodo1:~# sh -c "echo 'deb https://mirrors.up.pt/pub/mariadb/repo/10.10/debian bullseye main' >>
/etc/apt/sources.list"
root@nodo1:~#
```

Figura 3.8: Agregar repositorio de *MariaDB*

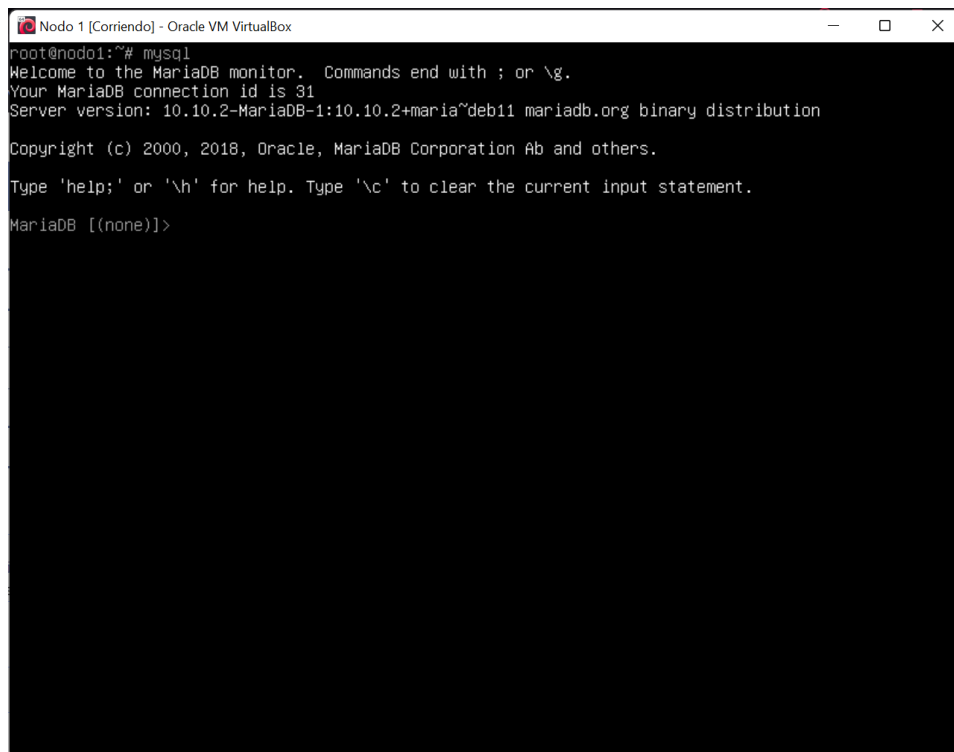
Una vez que ya tenemos lo necesario, instalamos *MariaDB*.



```
root@nodo1:~# apt update
Hit:1 http://security.debian.org/debian-security bullseye-security InRelease
Hit:2 http://deb.debian.org/debian bullseye InRelease
Hit:3 http://deb.debian.org/debian bullseye-updates InRelease
Hit:4 https://mirrors.up.pt/pub/mariadb/repo/10.10/debian bullseye InRelease
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
All packages are up to date.
root@nodo1:~# apt install mariadb-server
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  galera-4 gawk libcgi-fast-perl libcgi-pm-perl libclone-perl libconfig-inifiles-perl libdaxctl1
  libdbd-mariadb-perl libdbi-perl libencode-locale-perl libfcgi-bin libfcgi-perl libfcgi0ldbl
  libhtml-parser-perl libhtml-tagset-perl libhtml-template-perl libhttp-date-perl
  libhttp-message-perl libio-html-perl liblwp-mediatypes-perl libmariadb3 libndctl6 libpmem1
  libsigsegv2 libterm-readkey-perl libtimedate-perl liburi-perl liburing1 mariadb-client
  mariadb-client-core mariadb-common mariadb-server-core mysql-common psmisc pv rsync socat
Suggested packages:
  gawk-doc libmldbm-perl libnet-daemon-perl libsql-statement-perl libdata-dump-perl
  libipc-sharedcache-perl libwww-perl mailx mariadb-test netcat-openbsd doc-base
The following NEW packages will be installed:
  galera-4 gawk libcgi-fast-perl libcgi-pm-perl libclone-perl libconfig-inifiles-perl libdaxctl1
  libdbd-mariadb-perl libdbi-perl libencode-locale-perl libfcgi-bin libfcgi-perl libfcgi0ldbl
  libhtml-parser-perl libhtml-tagset-perl libhtml-template-perl libhttp-date-perl
  libhttp-message-perl libio-html-perl liblwp-mediatypes-perl libmariadb3 libndctl6 libpmem1
  libsigsegv2 libterm-readkey-perl libtimedate-perl liburi-perl liburing1 mariadb-client
  mariadb-client-core mariadb-common mariadb-server mariadb-server-core mysql-common psmisc pv
  rsync socat
0 upgraded, 38 newly installed, 0 to remove and 0 not upgraded.
Need to get 30.6 MB of archives.
After this operation, 252 MB of additional disk space will be used.
Do you want to continue? [Y/n] _
```

Figura 3.9: Instalación de *MariaDB*

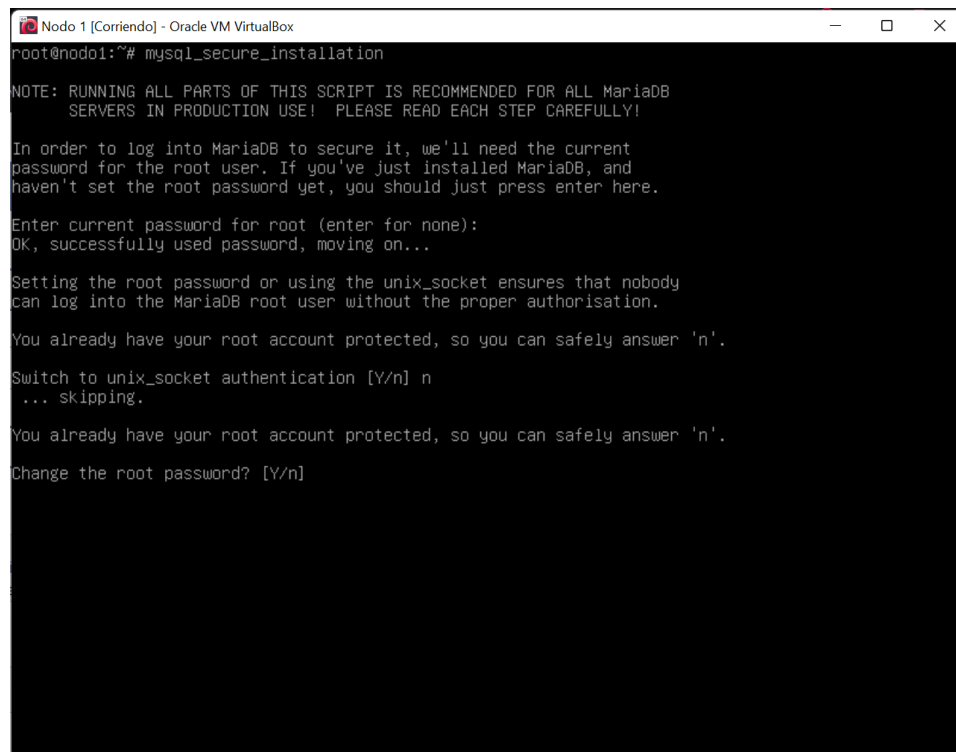
Una vez instalado *MariaDB*, comprobamos que se ha instalado correctamente.



```
Nodo 1 [Corriendo] - Oracle VM VirtualBox
root@nodo1:~# mysql
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 31
Server version: 10.10.2-MariaDB-1:10.10.2+maria~deb11 mariadb.org binary distribution
Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.
Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.
MariaDB [(none)]>
```

Figura 3.10: *MariaDB*

Habiendo comprobado que *MariaDB* se ha instalado correctamente, hacemos la instalación más segura con `mysql_secure_installation`.



```
Nodo 1 [Corriendo] - Oracle VM VirtualBox
root@nodo1:~# mysql_secure_installation

NOTE: RUNNING ALL PARTS OF THIS SCRIPT IS RECOMMENDED FOR ALL MariaDB
SERVERS IN PRODUCTION USE! PLEASE READ EACH STEP CAREFULLY!

In order to log into MariaDB to secure it, we'll need the current
password for the root user. If you've just installed MariaDB, and
haven't set the root password yet, you should just press enter here.

Enter current password for root (enter for none):
OK, successfully used password, moving on...

Setting the root password or using the unix_socket ensures that nobody
can log into the MariaDB root user without the proper authorisation.

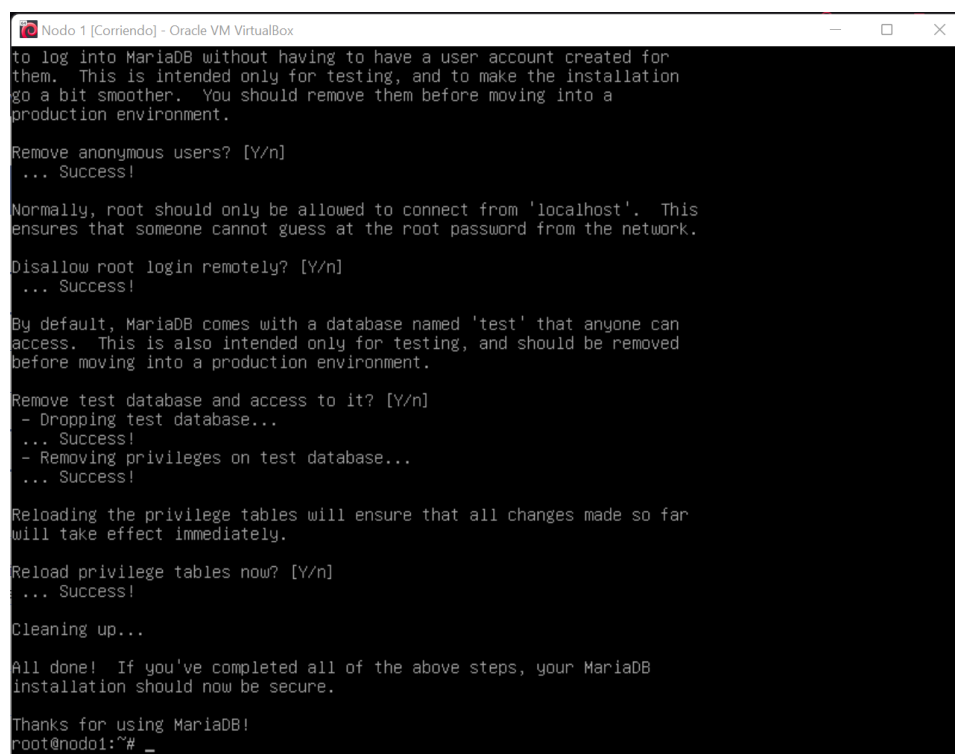
You already have your root account protected, so you can safely answer 'n'.

Switch to unix_socket authentication [Y/n] n
... skipping.

You already have your root account protected, so you can safely answer 'n'.

Change the root password? [Y/n]
```

Figura 3.11: `mysql_secure_installation`



```
Nodo 1 [Corriendo] - Oracle VM VirtualBox
to log into MariaDB without having to have a user account created for
them. This is intended only for testing, and to make the installation
go a bit smoother. You should remove them before moving into a
production environment.

Remove anonymous users? [Y/n]
... Success!

Normally, root should only be allowed to connect from 'localhost'. This
ensures that someone cannot guess at the root password from the network.

Disallow root login remotely? [Y/n]
... Success!

By default, MariaDB comes with a database named 'test' that anyone can
access. This is also intended only for testing, and should be removed
before moving into a production environment.

Remove test database and access to it? [Y/n]
- Dropping test database...
... Success!
- Removing privileges on test database...
... Success!

Reloading the privilege tables will ensure that all changes made so far
will take effect immediately.

Reload privilege tables now? [Y/n]
... Success!

Cleaning up...

All done! If you've completed all of the above steps, your MariaDB
installation should now be secure.

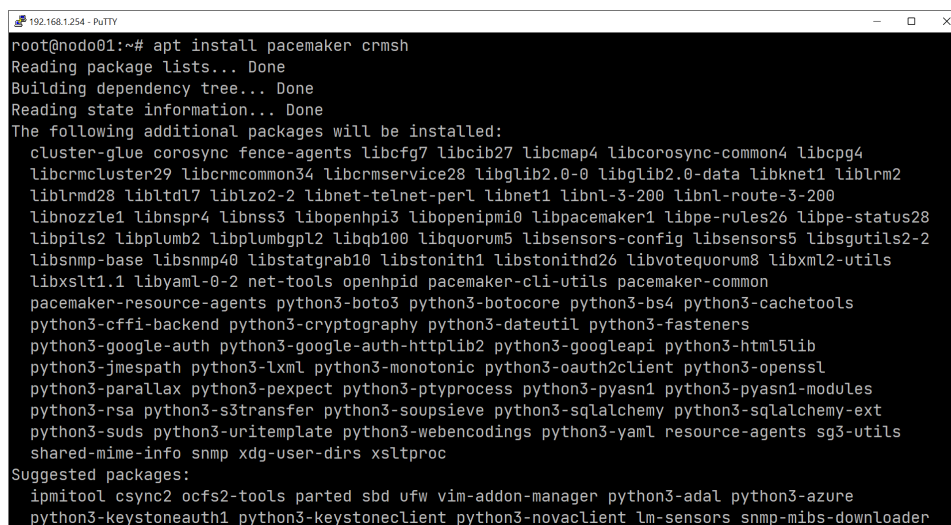
Thanks for using MariaDB!
root@nodo1:~# _
```

Figura 3.12: mysql_secure_installation

Cluster

En este capítulo, vamos a crear un cluster de alta disponibilidad para el servicio *Apache*. Esto nos va a permitir tener nuestra aplicación activa en caso de que el nodo principal falle por cualquier motivo. Para ello vamos a instalar las herramientas necesarias.

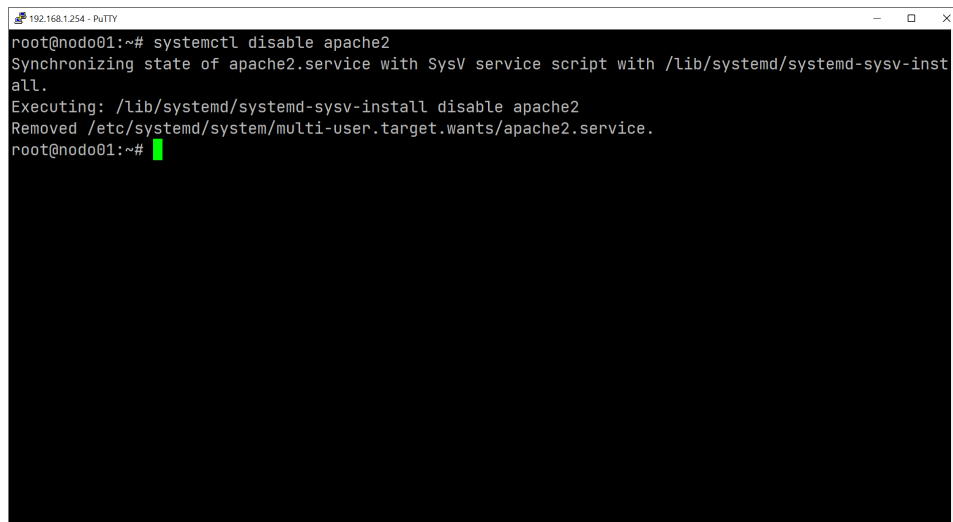
Empezamos instalando *pacemaker* y *crmsh*



```
root@nodo01:~# apt install pacemaker crmsh
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
cluster-glue corosync fence-agents libcfp7 libclib27 libcmmap4 libcorosync-common4 libcpq4
libcrmcluster29 libcrmcommon34 libcrmservice28 libglib2.0-0 libglib2.0-data libknet1 liblrm2
liblrm28 libltdl7 liblzo2-2 libnet-telnet-perl libnet1 libnl-3-200 libnl-route-3-200
libnozzle1 libnspr4 libnss3 libopenhpi3 libopenipmi0 libpacemaker1 libpe-rules26 libpe-status28
libpils2 libplumb2 libplumbgpl2 libqb100 libquorum5 libensors-config libensors5 libsgutils2-2
libsnmp-base libsnmp40 libstatgrab10 libstonith1 libstonithd26 libvotequorum8 libxml2-utils
libxslt1.1 libyaml-0-2 net-tools openhpid pacemaker-cli-utils pacemaker-common
pacemaker-resource-agents python3-boto3 python3-botocore python3-bs4 python3-cachetools
python3-cffi-backend python3-cryptography python3-dateutil python3-fasteners
python3-google-auth python3-google-auth-httpplib2 python3-googleapi python3-html5lib
python3-jmespath python3-lxml python3-monotonic python3-oauth2client python3-openssl
python3-parallax python3-pexpect python3-ptyprocess python3-pyasn1 python3-pyasn1-modules
python3-rsa python3-s3transfer python3-soupsieve python3-sqlalchemy python3-sqlalchemy-ext
python3-suds python3-uritemplate python3-webencodings python3-yaml resource-agents sg3-utils
shared-mime-info snmp xdg-user-dirs xsltproc
Suggested packages:
ipmitool csync2 ocfs2-tools parted sbd ufw vim-addon-manager python3-adal python3-azure
python3-keystoneauth1 python3-keystoneclient python3-novaclient lm-sensors snmp-mibs-downloader
```

Figura 4.1: Instalación de *pacemaker* y *crmsh*

Una vez instalado, tenemos que deshabilitar el inicio automático de *apache*, ya que lo controlará *pacemaker*.



```
root@nodo01:~# systemctl disable apache2
Synchronizing state of apache2.service with SysV service script with /lib/systemd/systemd-sysv-inst
all.
Executing: /lib/systemd/systemd-sysv-install disable apache2
Removed /etc/systemd/system/multi-user.target.wants/apache2.service.
root@nodo01:~#
```

Figura 4.2: Deshabilitar *apache*

A continuación, tenemos que editar el archivo *corosync.conf* para configurar el cluster. Tenemos que tener el mismo archivo de configuración en ambos nodos del cluster.

```
# Please read the corosync.conf.5 manual page
totem {
    version: 2
    protocol: udpu
    # Corosync itself works without a cluster name, but DLM needs
    ↪ one.
    # The cluster name is also written into the VG metadata of
    ↪ newly
    # created shared LVM volume groups, if lvmlockd uses DLM
    ↪ locking.
    cluster_name: flowribera

    # crypto_cipher and crypto_hash: Used for mutual node
    ↪ authentication.
    # If you choose to enable this, then do remember to create a
    ↪ shared
    # secret with "corosync-keygen".
```

```
# enabling crypto_cipher, requires also enabling of
↪ crypto_hash.
# crypto works only with knet transport

interface {
    ringnumber: 0
    bindnetaddr: 10.10.10.0
}
}

logging {
    # Log the source file and line where messages are being
    # generated. When in doubt, leave off. Potentially useful for
    # debugging.
    fileline: off
    # Log to standard error. When in doubt, set to yes. Useful
    ↪ when
    # running in the foreground (when invoking "corosync -f")
    to_stderr: yes
    # Log to a log file. When set to "no", the "logfile" option
    # must not be set.
    to_logfile: yes
    logfile: /var/log/corosync/corosync.log
    # Log to the system log daemon. When in doubt, set to yes.
    to_syslog: yes
    # Log debug messages (very verbose). When in doubt, leave off
    ↪ .
    debug: off
    # Log messages with time stamps. When in doubt, set to hires
    ↪ (or on)
    timestamp: on
    logger_subsys {
        subsys: QUORUM
        debug: off
    }
}

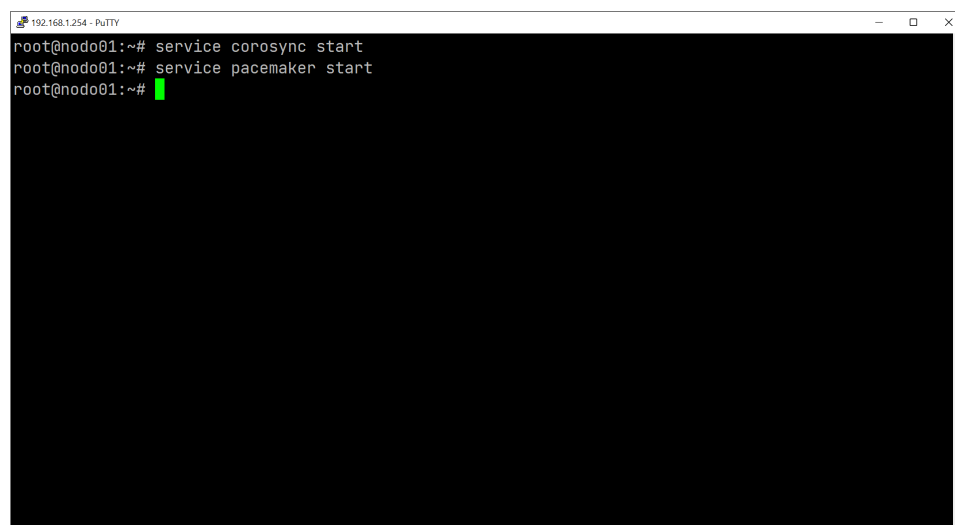
quorum {
    # Enable and configure quorum subsystem (default: off)
    # see also corosync.conf.5 and votequorum.5
    provider: corosync_votequorum
    two_nodes: 1
    expected_votes: 2
}

odelist {
    # Change/uncomment/add node sections to match cluster
    ↪ configuration
```

```
node {  
    # Hostname of the node  
    name: nodo01  
    # Cluster membership node identifier  
    nodeid: 1  
    # Address of first link  
    ring0_addr: 10.10.10.10  
    # When knet transport is used it's possible to define up to  
    ↪ 8 links  
    #ring1_addr: 192.168.1.1  
}  
  
node {  
    name: nodo02  
    nodeid: 2  
    ring0_addr: 10.10.10.20  
}  
# ...  
}
```

Código 4.1: corosync.conf

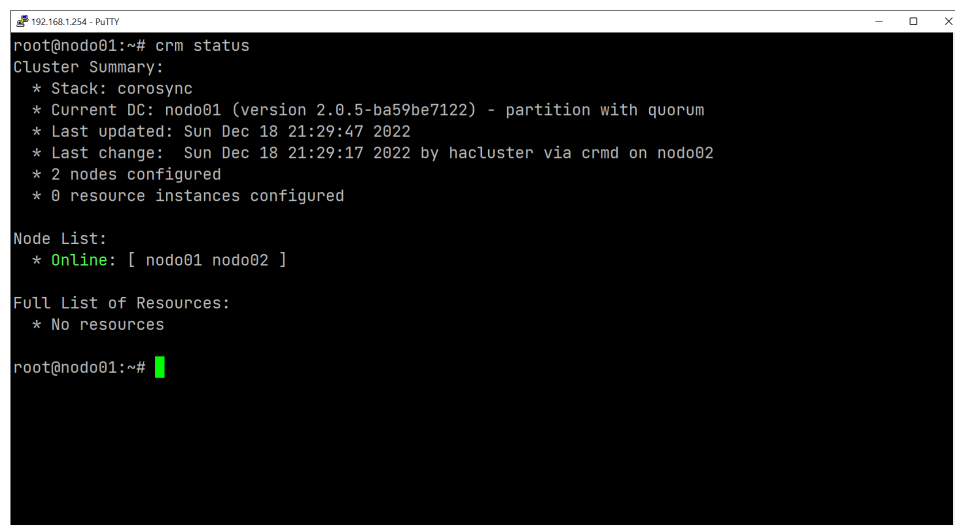
Iniciamos los servicios de *corosync* y *pacemaker* en ambos nodos.



```
192.168.1.254 - PuTTY  
root@nodo01:~# service corosync start  
root@nodo01:~# service pacemaker start  
root@nodo01:~#
```

Figura 4.3: Inicio de los servicios *corosync* y *pacemaker*

Si todo ha ido correctamente, al ejecutar el comando `crm status` debería salirnos lo siguiente.



```
root@nodo01:~# crm status
Cluster Summary:
* Stack: corosync
* Current DC: nodo01 (version 2.0.5-ba59be7122) - partition with quorum
* Last updated: Sun Dec 18 21:29:47 2022
* Last change: Sun Dec 18 21:29:17 2022 by hacluster via crmd on nodo02
* 2 nodes configured
* 0 resource instances configured

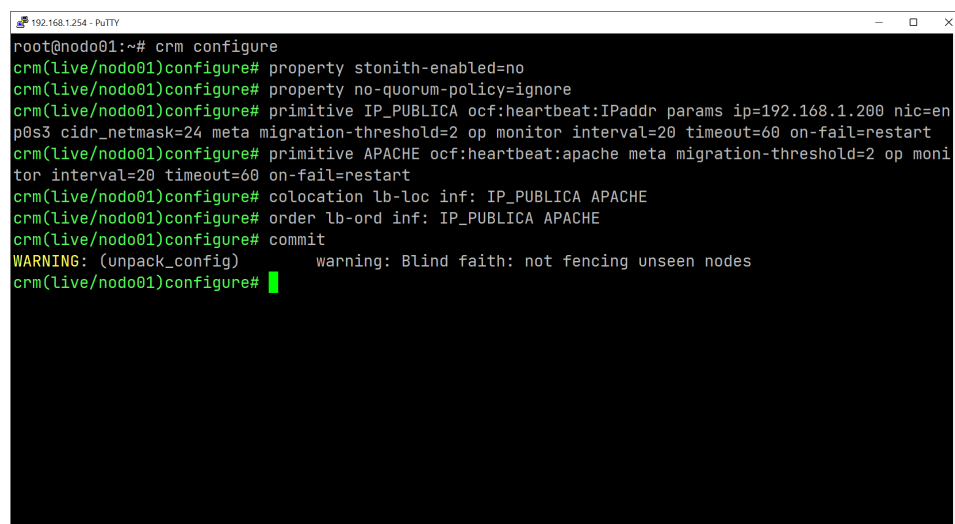
Node List:
* Online: [ nodo01 nodo02 ]

Full List of Resources:
* No resources

root@nodo01:~#
```

Figura 4.4: Estado del cluster

Ahora tenemos que configurar el cluster para que *apache* funcione en los 2 nodos. Para ello tenemos que ejecutar una serie de comandos bajo `crm configure`.



```
root@nodo01:~# crm configure
crm(live/nodo01)configure# property stonith-enabled=no
crm(live/nodo01)configure# property no-quorum-policy=ignore
crm(live/nodo01)configure# primitive IP_PUBLICA ocf:heartbeat:IPaddr params ip=192.168.1.200 nic=en
p0s3 cidr_netmask=24 meta migration-threshold=2 op monitor interval=20 timeout=60 on-fail=restart
crm(live/nodo01)configure# primitive APACHE ocf:heartbeat:apache meta migration-threshold=2 op moni
tor interval=20 timeout=60 on-fail=restart
crm(live/nodo01)configure# colocation lb-loc inf: IP_PUBLICA APACHE
crm(live/nodo01)configure# order lb-ord inf: IP_PUBLICA APACHE
crm(live/nodo01)configure# commit
WARNING: (unpack_config)      warning: Blind faith: not fencing unseen nodes
crm(live/nodo01)configure#
```

Figura 4.5: configuración del cluster