VIRTUAL CABLE

Configurar UDS Enterprise 3.6 en alta disponibilidad

UDS ENTERPRISE 3.6

Índice

Introducción 2
Elementos necesarios 3
Servidores de base de datos MySQL 3
Servidores HAProxy 3
Servidores UDS-Server 4
Servidores UDS-Tunneler 4
Requisitos para el despliegue 5
Configuración de los servidores MySQL 6
Configuración del clúster galera con 3 servidores de base de datos. 6
Verificación de réplica de base de datos 12
Configuración de los servidores HAProxy 13
Instalando HAProxy en Linux Debian 22
Configuración de los servidores UDS Server y Tunnel 40
Configuración servidores UDS (UDS-Server) 40
Configuración servidores Tunnel (UDS-Tunnel) 47
Respuesta ante caída del servidor HA Proxy Maestro 54
Tareas de recuperación del clúster de Galera 55
El nodo 1 se detiene correctamente 55
Dos nodos se detienen correctamente 55
Los tres nodos se detienen correctamente 56
Un nodo desaparece del clúster 57
Dos nodos desaparecen del clúster 57
Todos los nodos se caen sin un procedimiento de apagado adecuado 58
El cluster pierde su estado primario debido a la “división del cerebro” 60
Sobre Virtual Cable 61

Introducción

UDS Enterprise permite realizar la configuración de sus diferentes componentes en alta disponibilidad (HA). Este modo de configuración permite dotar al entorno VDI de continuidad ante el fallo de algún nodo de virtualización o por al fallo del propio S.O. de alguno de los componentes del entorno.

Para dotar al entorno VDI de una alta disponibilidad completa, además de configurar varias máquinas UDS-Server y UDS-Tunnel, también será necesario disponer de una replicación o configuración en cluster de la base de datos a la que se conectan los servidores UDS. Otro elemento necesario y que también tendremos que configurar en alta disponibilidad, será el balanceador de carga que gestione y reparta las diferentes conexiones a los componentes UDS-Server y UDS-Tunnel y nuestro cluster de base de datos.

UDS Enterprise soporta balanceadores de tipo físico (ej: F5) o de tipo virtual (ej: HAProxy), estos tienen que tener soporte para modos TCP y HTTP.

En el presente documento, a través de un ejemplo completo de configuración, trataremos de abordar todos los pasos para configurar UDS Enterprise en Alta Disponibilidad, desde los elementos propios de UDS (UDS-Server y UDS-Tunnel) hasta un balanceador de carga software (HAProxy) y una Base de datos MySQL configurada con un cluster galera.

Elementos necesarios

En esta guía utilizaremos los componentes necesarios para la mayoría de los despliegues de un entorno UDS en HA. Son los siguientes:

Servidores de base de datos MySQL

Los servidores de base de datos (BBDD) que utilizaremos serán los proporcionados por el equipo de UDS. En estos servidores se guardarán todos los registros y configuraciones de UDS.

En este documento mostramos la configuración de tres servidores MySQL en modo de replicación activo/activo.

NOTA:
A partir de la versión 3.0 de UDS Enterprise, se soportan configuraciones de clusters MySQL activo/activo.
El componente de base de datos es uno de los componentes más importantes del entorno VDI con UDS. Por tanto, para despliegues en producción se recomienda encarecidamente disponer de respaldo en este componente, ya sea vía backup de máquina completa, instancia de BD utilizada en UDS, configuración en cluster, o como se mostrará en este documento, una configuración de réplica activo/activo.

Servidores HAProxy

Será el servidor encargado de balancear las conexiones de los servidores UDS Server y Tunnel. A través de él se realizará el acceso de usuarios/administradores en el portal de login de UDS y las conexiones a los diferentes servicios.

En este documento se configuran dos máquinas HAProxy, en modo activo/pasivo.

NOTA:
En los diferentes servidores HAProxy configuraremos una dirección IP que estará activa solamente en el servidor principal.En caso de caída o aislamiento de este servidor, se activará automáticamente en los otros servidores secundarios HAProxy.

Servidores UDS-Server

Podremos añadir todas las máquinas UDS-Server que necesitemos y hacerlas funcionar en modo activo/activo. Esto permitirá acceso continuo al portal de login a usuarios y administradores, aunque perdamos alguna de las máquinas UDS-Server.

En este documento se configuran dos máquinas UDS-Server, en modo activo/activo.

Servidores UDS-Tunneler

Podremos añadir todas las máquinas UDS-Tunnel que necesitemos y hacerlas funcionar en modo activo/activo, esto permitirá acceso a servicios (escritorios o aplicaciones) a través de conexiones tunelizadas y HTML5 aunque perdamos alguna de las máquinas UDS-Tunnel.

En este documento se configuran dos máquinas UDS-Tunnel, en modo activo/activo.

NOTA:
Si un usuario está conectado a un servicio (escritorio o aplicación) y cae el servidor tunnel por el que está conectado, la conexión se perderá. Pero al volver a realizar la conexión, recuperará acceso al servicio a través de otro servidor tunnel activo de forma automática.

Requisitos para el despliegue

En este ejemplo de configuración de UDS Enterprise en HA, se han utilizado los siguientes recursos:

MySQL:

- 3 servidores MySQL (proporcionados por el equipo de UDS Enterprise). Los requisitos mínimos para cada máquina son: 2 vCPUs, 1 GB de vRAM y 10 GB de disco
- Datos IP: Al menos 3 direcciones IP, una para cada servidor, máscara de red, Gateway y DNS.
- Datos BBDD: Instancia, usuario y contraseña (por defecto, instancia: uds, usuario: uds, contraseña: uds).
- **Habrá que tener en cuenta que este appliance no cuenta con soporte directo por parte de UDS Enterprise**

HAProxy:

- 2 máquinas con S.O. Linux Debian (puede utilizar servidores preconfigurados proporcionados por UDS disponibles en este repositorio: https://images.udsenterprise.com/files/UDS_HA/HAProxy/3.6/OVA-3.6/ con al menos 2 vCPUs, 1 GB de vRAM, 10 GB de disco.
- Datos IP: 3 direcciones IP, una para cada servidor (Master - Slave) y una IP virtual compartida entre los dos servidores que servirá para el balanceo), máscara de red, gateway y DNS.
- Acceso a internet.
- Certificado: Es necesario disponer (o generar) un certificado válido para las conexiones SSL en formato PEM. En este ejemplo se muestra cómo crear un certificado temporal.

UDS-Server:

- 2 máquinas UDS-Server (proporcionados por el equipo de UDS Enterprise). Los requisitos mínimos por cada máquina son: 2 vCPUs, 2 GB de vRAM y 8 GB de disco.
- Datos IP: 2 direcciones IP, una para cada servidor, máscara de red, gateway y DNS.
- Número de serie válido.
- Datos de conexión con la BBDD MySQL: dirección IP, instancia, usuario y contraseña.

UDS-Tunnel:

- 2 máquinas UDS-Tunnel (proporcionados por el equipo de UDS Enterprise). Los requisitos mínimos por cada máquina son: 2 vCPUs, 2 GB de vRAM y 14 GB de disco.
- Datos IP: 2 direcciones IP, una para cada servidor, máscara de red, gateway y DNS.
- Dirección IP de balanceo de los servidores HAProxy.

Configuración de los servidores MySQL

Para poder realizar la conexión de UDS con un clúster MariaDB, necesitaremos los siguientes componentes:

- 3 o más máquinas con un servidor MariaDB instalado. (En este caso se realizará con la máquina de base de datos de ejemplo facilitada por el equipo de UDS Enterprise)

Nodo 1: 192.168.1.69

Nodo 2: 192.168.1.67

Nodo 3: 192.168.1.70

- Tener 1 o más UDS Server en la plataforma para poder hacer la conexión con la base de datos.

Configuración del clúster galera con 3 servidores de base de datos.

En nuestras 3 máquinas tendremos que realizar una actualización de paquetes:

Sudo apt update -y

Sudo apt upgrade -y

Es importante configurar todas las **Ips** de los nodos que vayamos a utilizar, la Ip del nodo a configurar, así como su **hostname** personalizado (comando: hostnamectl set-hostname --static *nombre_servidor*). Esto lo realizamos en todos los nodos según su configuración de red.

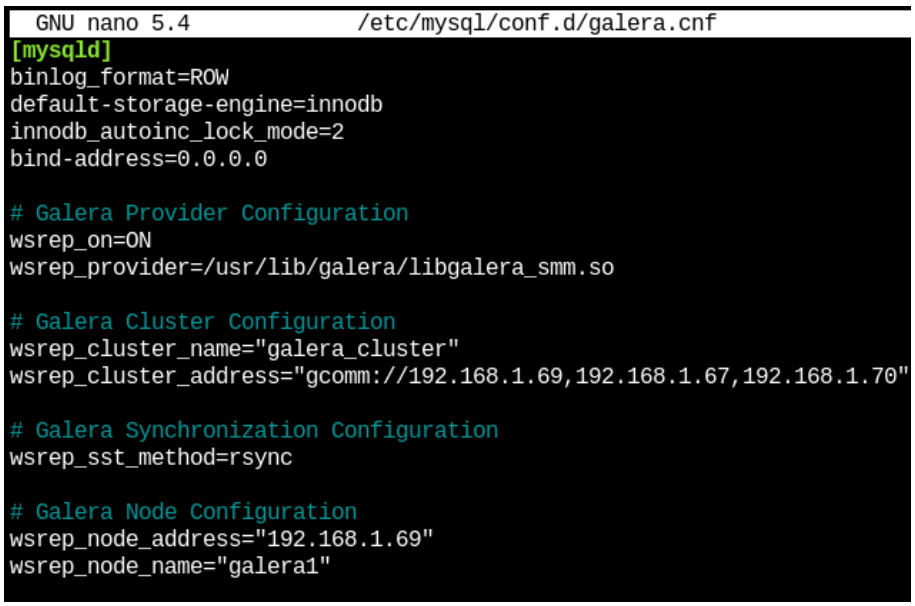
Para comenzar Crearemos el archivo de configuración del cluster en cada nodo en la ruta:

/etc/mysql/conf.d/galera.cnf

Introduciremos en cada nodo la información necesaria para el correcto funcionamiento del cluster:

En el primer nodo, este será el formato a seguir.

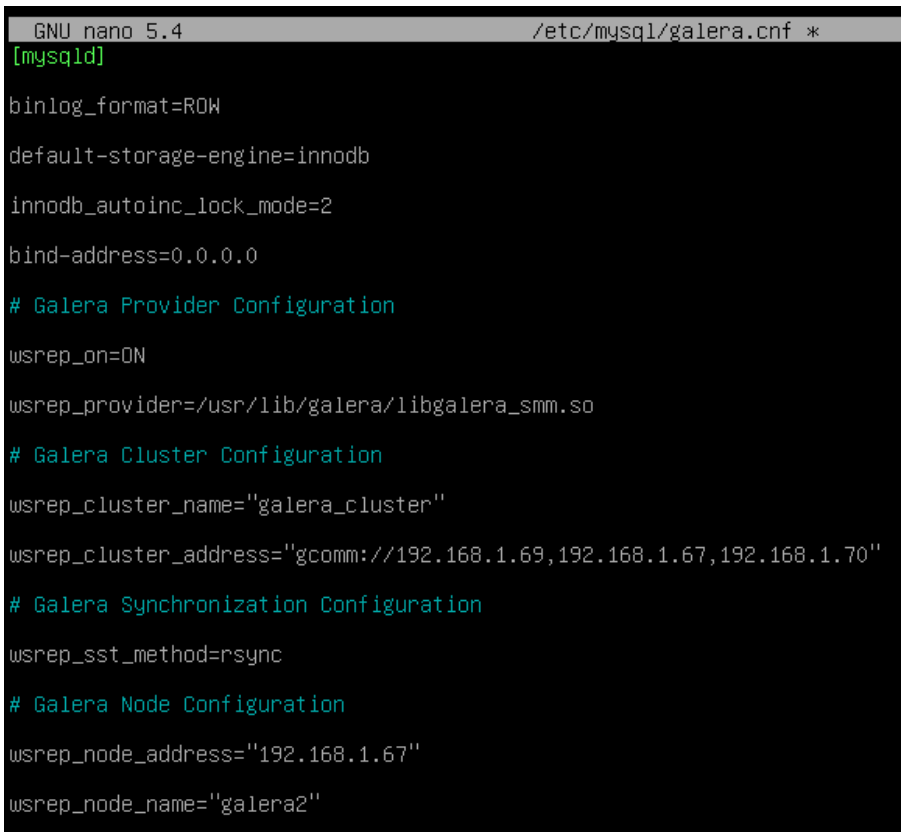
```
[mysqld]
binlog_format=ROW
default-storage-engine=innodb
innodb_autoinc_lock_mode=2
bind-address=0.0.0.0

# Galera Provider Configuration
wsrep_on=ON
wsrep_provider=/usr/lib/galera/libgalera_smm.so

# Galera Cluster Configuration
wsrep_cluster_name="galera_cluster"
wsrep_cluster_address="gcomm://node1-ip-address,node2-ip-address,node3-ip-address"

# Galera Synchronization Configuration
wsrep_sst_method=rsync

# Galera Node Configuration
wsrep_node_address="node1-ip-address"
wsrep_node_name="node1"
```



Nodo1

Nodo número 2, en la misma ruta que el anterior crearemos el archivo con el siguiente formato

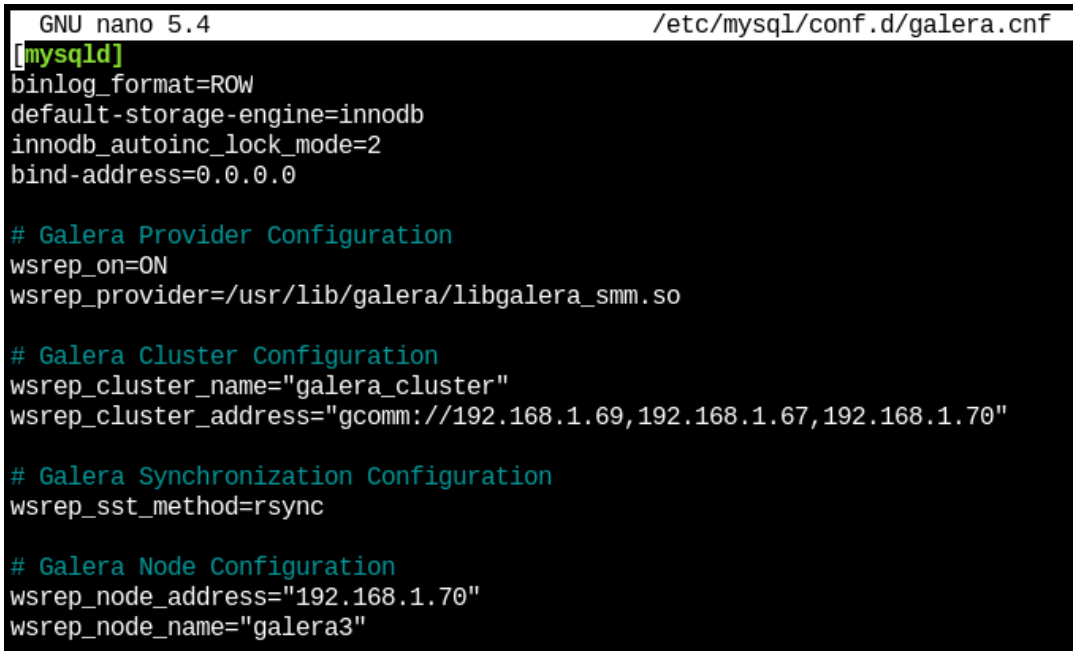
```
[mysqld]
binlog_format=ROW
default-storage-engine=innodb
innodb_autoinc_lock_mode=2
bind-address=0.0.0.0

# Galera Provider Configuration
wsrep_on=ON
wsrep_provider=/usr/lib/galera/libgalera_smm.so

# Galera Cluster Configuration
wsrep_cluster_name="galera_cluster"
wsrep_cluster_address="gcomm://node1-ip-address,node2-ip-address,node3-ip-address"

# Galera Synchronization Configuration
wsrep_sst_method=rsync

# Galera Node Configuration
wsrep_node_address="node2-ip-address"
wsrep_node_name="node2"
```



Nodo2

Nodo número 3, en la misma ruta que el anterior crearemos el archivo con el siguiente formato

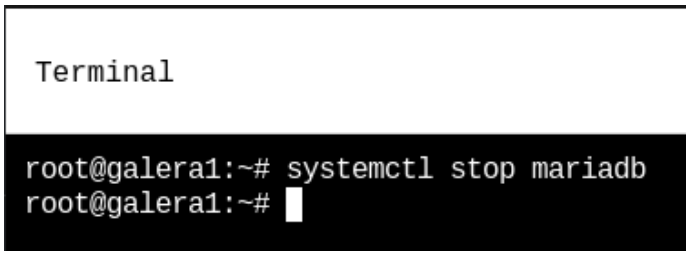
```
[mysqld]

binlog_format=ROW
default-storage-engine=innodb
innodb_autoinc_lock_mode=2
bind-address=0.0.0.0

# Galera Provider Configuration
wsrep_on=ON
wsrep_provider=/usr/lib/galera/libgalera_smm.so

# Galera Cluster Configuration
wsrep_cluster_name="galera_cluster"
wsrep_cluster_address="gcomm://node1-ip-address,node2-ip-address,node3-ip-address"

# Galera Synchronization Configuration
wsrep_sst_method=rsync

# Galera Node Configuration
wsrep_node_address="node3-ip-address"
wsrep_node_name="node3"
```



Nodo3

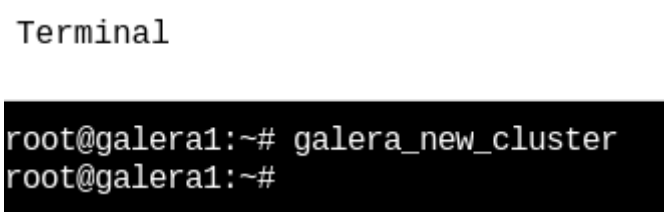
Después de configurar el archivo en todos los nodos de nuestra plataforma, tendremos que detener el servicio *mariadb* en **todos los nodos** para poder inicializar el clúster.

systemctl stop mariadb



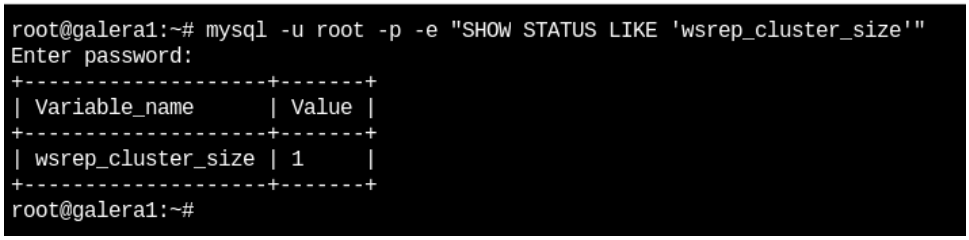
En el primer nodo, se inicializa el clúster MariaDB Galera con el siguiente comando:

galera_new_cluster



Ahora podremos ver el estado del clúster con el siguiente comando*:*

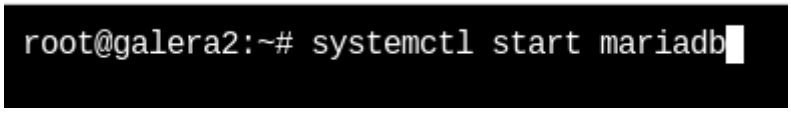
```
mysql -u root -p -e "SHOW STATUS LIKE 'wsrep_cluster_size'"
```



Se deberá ver el número "1".

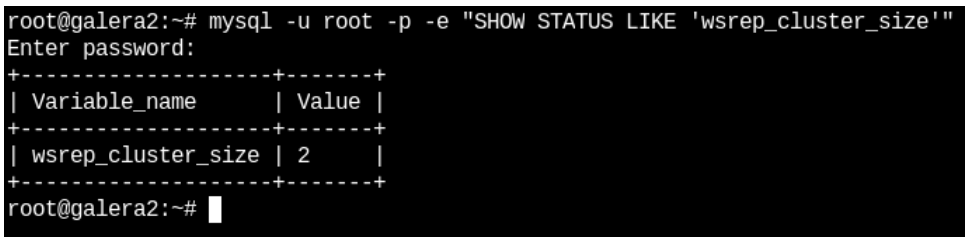
A continuación, en el segundo nodo se iniciará el servicio mariadb:

systemctl start mariadb



Ahora podremos volver a ver el estado del cluster con el siguiente comando (En el nodo 2):

```
mysql -u root -p -e "SHOW STATUS LIKE 'wsrep_cluster_size'"
```

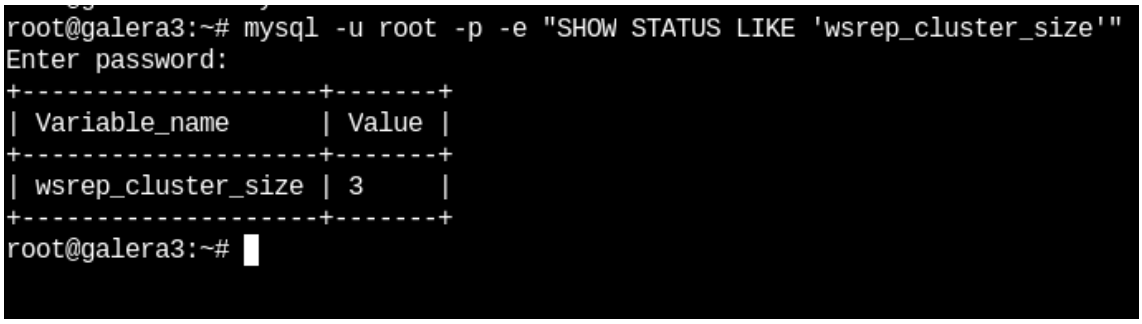


A continuación, en el tercer nodo se iniciará el servicio mariadb:

systemctl start mariadb

Ahora podremos volver a ver el estado del clúster con el siguiente comando (En el nodo 3):

```
mysql -u root -p -e "SHOW STATUS LIKE 'wsrep_cluster_size'"
```



Como podemos ver, los 3 nodos están activos y conectados entre sí

Verificación de réplica de base de datos

A continuación, se va a verificar la replicación de las bases de datos.

En el primero nodo conectaremos con maríadb:

mysql -u root -p

Una vez dentro crearemos unas bases de datos con el siguiente comando:

```
Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MariaDB [(none)]> create database db1;
Query OK, 1 row affected (0.002 sec)
```

En el nodo 2 y 3 verificaremos que estas bases de datos existen

```
MariaDB [(none)]> show databases;
+--------------------+
| Database           |
+--------------------+
| db1                |
| information_schema |
| mysql              |
| performance_schema |
| uds                |
+--------------------+
5 rows in set (0.000 sec)

MariaDB [(none)]>
```

```
root@galera3:~# mysql -u root
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 36
Server version: 10.5.15-MariaDB-0+deb11u1 Debian 11

Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input

MariaDB [(none)]> show databases;
+--------------------+
| Database           |
+--------------------+
| db1                |
| information_schema |
| mysql              |
| performance_schema |
| uds                |
+--------------------+
5 rows in set (0.000 sec)

MariaDB [(none)]>
```

De esta manera hemos comprobado que toda la información que se almacene en la primera base de datos se replicará automáticamente en el resto.

Configuración de los servidores HAProxy

En este documento se utilizarán los servidores HAProxy facilitados por el equipo de UDS Enterprise. Estos servidores están preconfigurados y solo será necesario modificar ciertos datos para tenerlos completamente configurados.

Los servidores los podremos descargar del siguiente repositorio:

https://images.udsenterprise.com/files/UDS_HA/HAProxy/3.6/OVA-3.6/

Ambos servidores están configurados con los siguientes recursos: 2 vCPUs, 1 GB de vRAM, 10 GB de disco y 1 vNIC.

Los servidores tienen un usuario creado: ***user***, con la contraseña: ***uds***. La contraseña del usuario root es: ***uds***

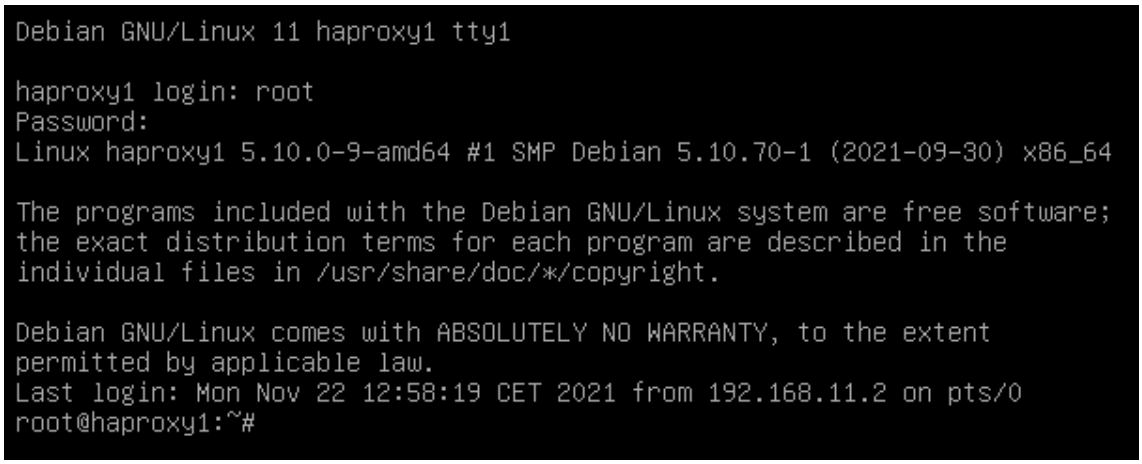
Una vez importados a la plataforma de virtualización, procederemos a su configuración

NOTA:
Estos servidores se facilitan en formato .OVA preparados para importar en entornos VMware. Si fuera necesario importarlos en otra plataforma de virtualización diferente, se puede extraer (ej: Winrar) su disco. vmdk y convertir (ej: qemu.img) al formato de la plataforma destino.

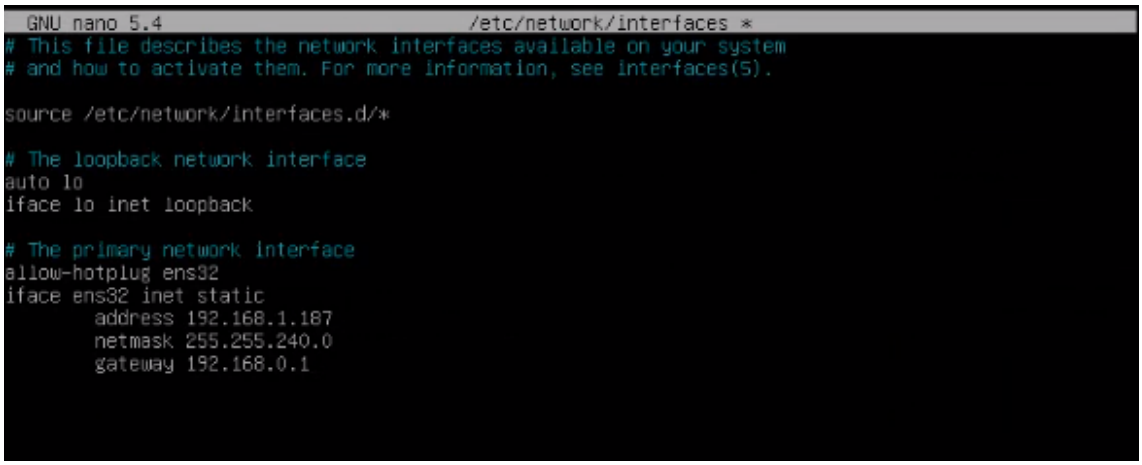
Se recomienda encarecidamente modificar la contraseña por defecto por una de mayor seguridad.

- **TAREAS A REALIZAR EN EL SERVIDOR HAPROXY PRINCIPAL**

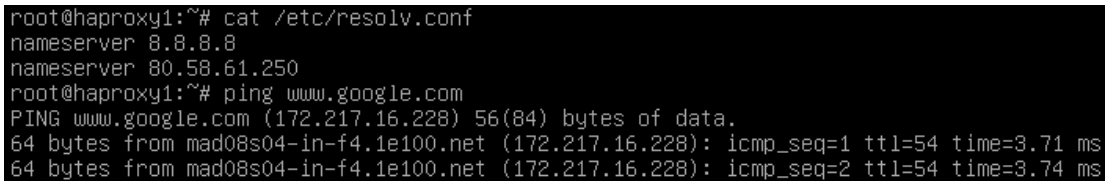
Una vez importada la máquina a la plataforma virtual y encendida, deberemos validarnos con el usuario: ***root*** y la contraseña: ***uds***



Configuraremos los nuevos datos IP modificando el fichero: /etc/network/interfaces



Y confirmaremos que tenemos unos datos DNS válidos y que disponemos de salida a internet.



Reiniciaremos el servidor para aplicar la nueva configuración IP.

Primero debemos ejecutar los comandos de actualización por si hubiera parches importantes se seguridad y de otros componentes que podamos aplicar:

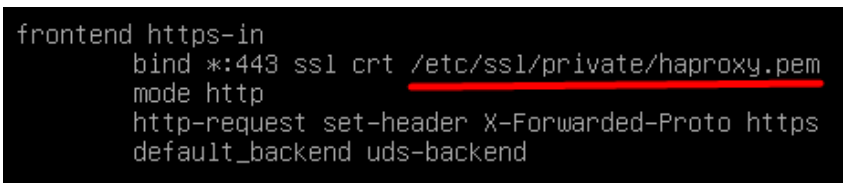
```
apt-get update
```

```
apt-get upgrade
```

Ahora procedemos a modificar los datos configurados en el servicio HAProxy. Para ello editaremos el fichero: **/etc/haproxy/haproxy.cfg**

En este documento solo se hará referencia a algunos parámetros. Se recomienda revisar a fondo el resto de los parámetros preconfigurados y modificarlos en base a las necesidades de cada entorno.

El servicio está preconfigurado con un certificado temporal autogenerado:



Regla acceso Frontend al servidor UDS en modo http (indicaremos la ruta del certificado. pem generado anteriormente). Puerto 443

```
frontend https-in
        bind *:443 ssl crt /etc/ssl/private/haproxy.pem
        mode http
        http-request set-header X-Forwarded-Proto https
        default_backend uds-backend
```

Regla acceso Frontend al servidor Tunnel en modo TCP por el **puerto 1443** (conexiones tunelizadas). En caso de utilizar otro puerto diferente será necesario modificarlo (este puerto es el que ha sido indicado en la pestaña Tunnel de un transporte vía tunnel).

```
frontend tunnel-in
        bind *:1443
        mode tcp
        option tcplog
        default_backend tunnel-backend-ssl
```

Regla acceso Frontend al servidor Tunnel en modo TCP por el **puerto 10443** (conexiones HTML5). En caso de utilizar otro puerto diferente será necesario modificarlo (este puerto es el que ha sido indicado en la pestaña Tunnel de un transporte HTML5).

```
frontend tunnel-in-guacamole    # HTML5
        bind *:10443
        mode tcp
        option tcplog
        default_backend tunnel-backend-guacamole
```

Regla de acceso backend al servidor UDS. **Deberemos indicar las direcciones IP de nuestras máquinas UDS-Server** (los puertos de escucha del servidor UDS son el 80 o el 443).

```
backend uds-backend
        option http-keep-alive
        balance source
        server udss1 192.168.0.183:443  check inter 2000 rise 2 fall 5 ssl verify none
        server udss2 192.168.0.184:443 check inter 2000 rise 2 fall 5 ssl verify none
```

Regla de acceso backend al servidor Tunnel para las conexiones tunelizadas. **Deberemos indicar las direcciones IP de nuestras máquinas UDS-Tunnel** (el puerto de escucha del servidor Tunnel para las conexiones tunelizadas es 443).

```
backend tunnel-backend-ssl
        mode tcp
        option tcplog
        balance roundrobin
        server udst1 192.168.1.185:443 check inter 2000 rise 2 fall 5
        server udst2 192.168.1.186:443 check inter 2000 rise 2 fall 5
```

Regla de acceso backend al servidor Tunnel para las conexiones HTML5. **Deberemos indicar las direcciones IP de nuestras máquinas UDS-Tunnel** (el puerto de escucha del servidor Tunnel para las conexiones HTML5 es 10443).

```
backend tunnel-backend-guacamole
        mode tcp
        option tcplog
        balance source
        server udstg1 192.168.1.185:10443 check inter 2000 rise 2 fall 5
        server udstg2 192.168.1.186:10443 check inter 2000 rise 2 fall 5
```

Reglas de acceso a las diferentes bases de datos, deberemos indicar las diferentes ips de nuestros servidores de base de datos (el puerto de escucha para las conexiones con las bases de datos es 3306)

```
frontend galera_cluster_frontend
        bind *:3306
        mode tcp
        option tcplog
        default_backend galera_cluster_backend

backend galera_cluster_backend
        mode tcp
        option tcpka
        balance source
        server db1 192.168.0.180:3306 check weight 1
        server db2 192.168.0.181:3306 check weight 2
        server db3 192.168.0.182:3306 check weight 3
```

frontend galera_cluster_frontend

bind *:3306

mode tcp

option tcplog

default_backend galera_cluster_backend

backend galera_cluster_backend

mode tcp

option tcpka

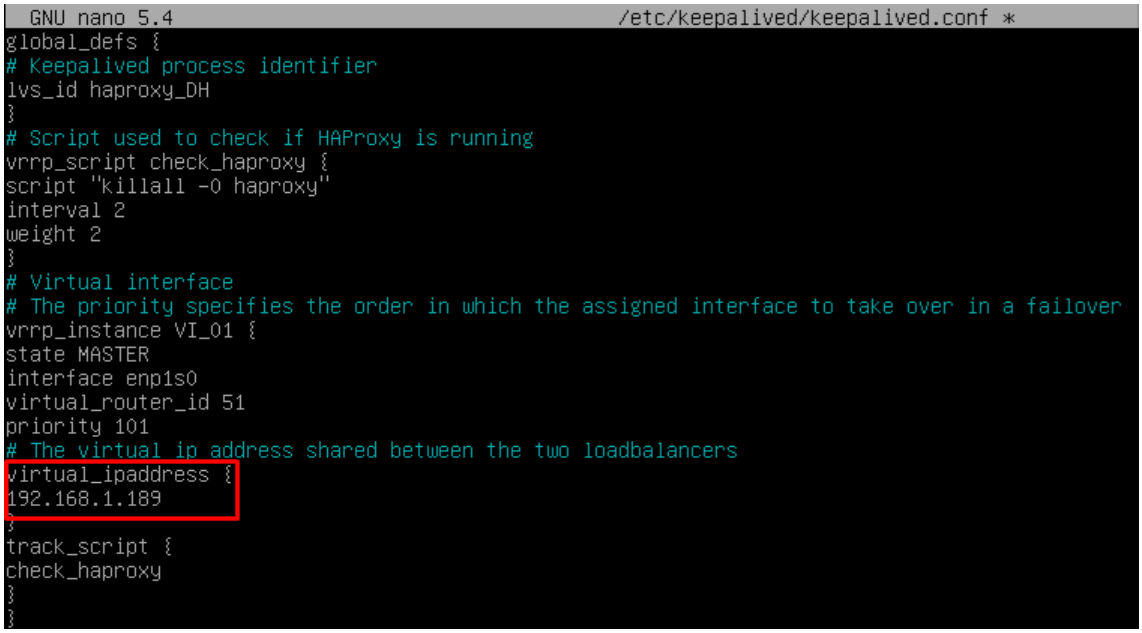
balance source

server db1 192.168.0.180:3306 check weight 1

server db2 192.168.0.181:3306 check weight 2

server db3 192.168.0.182:3306 check weight 3

Por último, indicaremos la IP virtual de balanceo que tendrán los servidores principal y secundario. Para ello editamos el fichero: /etc/keepalived/keepalived.conf



En este fichero también deberemos confirmar que el interfaz de red es el correcto (se puede confirmar con el comando `ip a`) y que el "rol" asignado será el de servidor principal (Master):

```
 GNU nano 5.4                                  /etc/keepalived/keepalived.conf *
global_defs {
# Keepalived process identifier
lvs_id haproxy_DH
}
# Script used to check if HAProxy is running
vrrp_script check_haproxy {
script "killall -0 haproxy"
interval 2
weight 2
}
# Virtual interface
# The priority specifies the order in which the assigned interface to take over in a failover
vrrp_instance VI_01 {
state MASTER
interface enp1s0
virtual_router_id 51
priority 101
# The virtual ip address shared between the two loadbalancers
virtual_ipaddress {
192.168.1.189
}
track_script {
check_haproxy
}
}
```

Reiniciaremos el servidor para aplicar todos los cambios y comprobaremos que la IP virtual de balanceo esta activa:

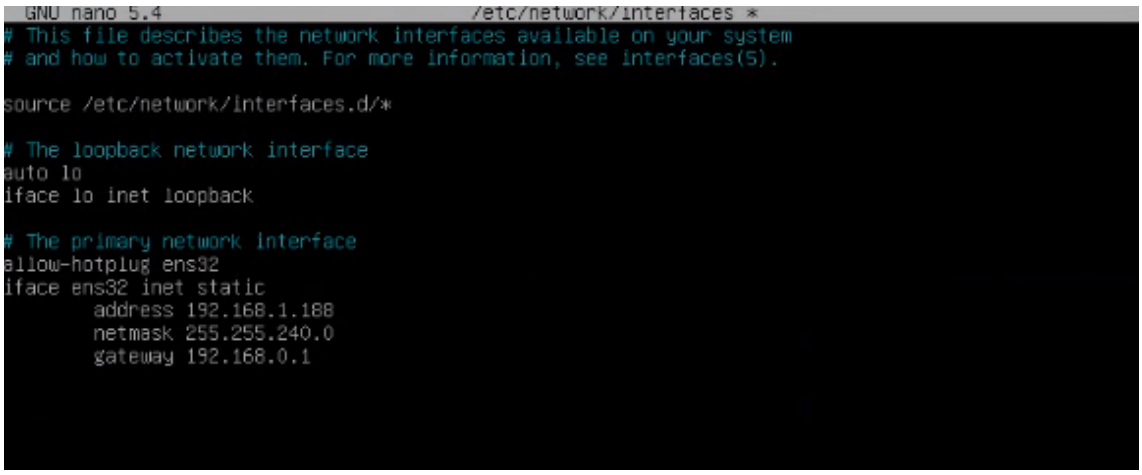
```
root@haproxy1:~# ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: enp1s0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP group default qlen 1000
    link/ether 52:54:00:99:e0:c5 brd ff:ff:ff:ff:ff:ff
    inet 192.168.1.187/20 brd 192.168.15.255 scope global enp1s0
       valid_lft forever preferred_lft forever
    inet 192.168.1.189/32 scope global enp1s0
       valid_lft forever preferred_lft forever
    inet6 fe80::5054:ff:fe99:e0c5/64 scope link
       valid_lft forever preferred_lft forever
root@haproxy1:~# _
```

NOTA:

La dirección IP virtual de balanceo será la que nos proporcione acceso al entorno UDS. Esta dirección permanecerá siempre activa en el servidor principal y, cuando esta sufra una caída, automáticamente se activará en el servidor secundario.

- **TAREAS A REALIZAR EN EL SERVIDOR HAPROXY SECUNDARIO**

Las tareas a realizar serán exactamente las mismas que en el servidor principal, indicaremos sus datos IP:



Reiniciaremos el servidor para aplicar la nueva configuración IP.

Ejecutaremos los comandos de actualización por si hubiera parches importantes se seguridad y de otros componentes que podamos aplicar:

```
apt-get update
```

```
apt-get upgrade
```

Modificar los mismos datos configurados en el servicio HAProxy que en el servidor principal (principalmente las direcciones IPs de los servidores UDS y Tunnel), editando el fichero: /etc/haproxy/haproxy.cfg

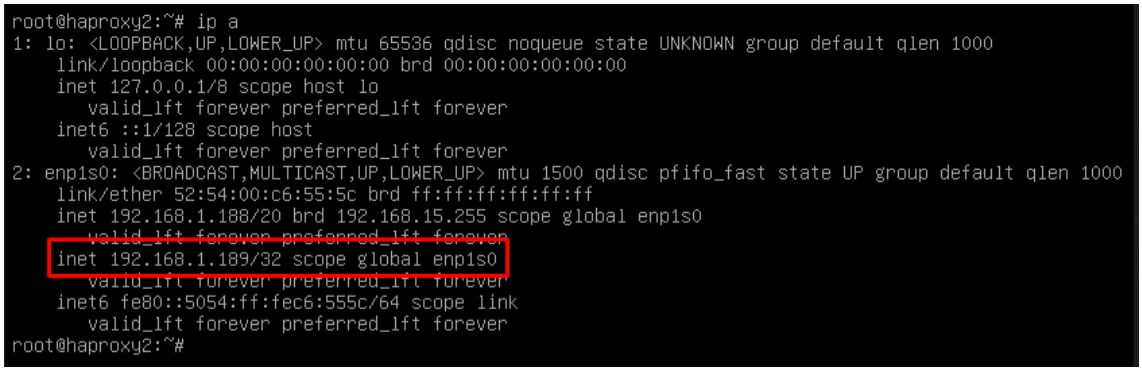
Por último, indicaremos la IP virtual de balanceo que tendrán los servidores principal y secundario, editando el fichero: /etc/keepalived/keepalived.conf

```
  GNU nano 5.4                                  /etc/keepalived/keepalived.conf *
global_defs {
# Keepalived process identifier
lvs_id haproxy_DH_passive
}
# Script used to check if HAProxy is running
vrrp_script check_haproxy {
script "killall -0 haproxy"
interval 2
weight 2
}
# Virtual interface
# The priority specifies the order in which the assigned interface to take over in a failover
vrrp_instance VI_01 {
state SLAVE
interface enp1s0
virtual_router_id 51
priority 100
# The virtual ip address shared between the two loadbalancers
virtual_ipaddress {
192.168.1.189
}
track_script {
check_haproxy
}
}
```

Y el único cambio significativo que tendrá el servidor secundario, además de confirmar que el interfaz de red es el correcto, será que el "rol" asignado al servidor secundario tiene que ser SLAVE:

```
  GNU nano 5.4                                  /etc/keepalived/keepalived.conf *
global_defs {
# Keepalived process identifier
lvs_id haproxy_DH_passive
}
# Script used to check if HAProxy is running
vrrp_script check_haproxy {
script "killall -0 haproxy"
interval 2
weight 2
}
# Virtual interface
# The priority specifies the order in which the assigned interface to take over in a failover
vrrp_instance VI_01 {
state SLAVE
interface enp1s0
virtual_router_id 51
priority 100
# The virtual ip address shared between the two loadbalancers
virtual_ipaddress {
192.168.1.189
}
track_script {
check_haproxy
}
}
```

Reiniciaremos el servidor para aplicar todos los cambios y, en este caso, comprobaremos que la IP virtual de balanceo no está activa. Solo se activará en caso de caída del servidor principal:



Instalando HAProxy en Linux Debian

Aunque en este documento se utilicen los servidores HAProxy preconfigurados y facilitados por el equipo de UDS Enterprise, también es posible su instalación y configuración completa partiendo de un S.O. nuevo.

En este apartado, mostraremos un ejemplo de su instalación y configuración completa sobre un S.O. Linux Debian. Utilizaremos unos recursos básicos: 2 vCPUs, 1 GB de vRAM, 8 GB de disco y 1 vNic.

Se mostrará la configuración del nodo primario. La mayoría de las tareas será necesarios realizarlas también en el nodo primario, exceptuando la generación del certificado, que solo se deberá generar en uno de los servidores, y la configuración del componente Keepalived, que en el caso del servidor secundario utilizará el modo Slave.

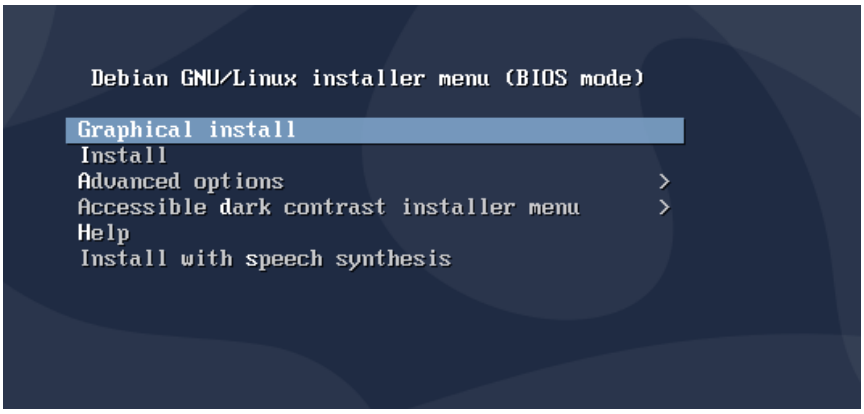
NOTA:
Si ya ha desplegado las máquinas HAProxy preconfiguradas y facilitadas por el equipo de UDS Enterprise, puede saltarse este apartado.

En esta instalación Instalaremos un S.O. Linux Debian 11

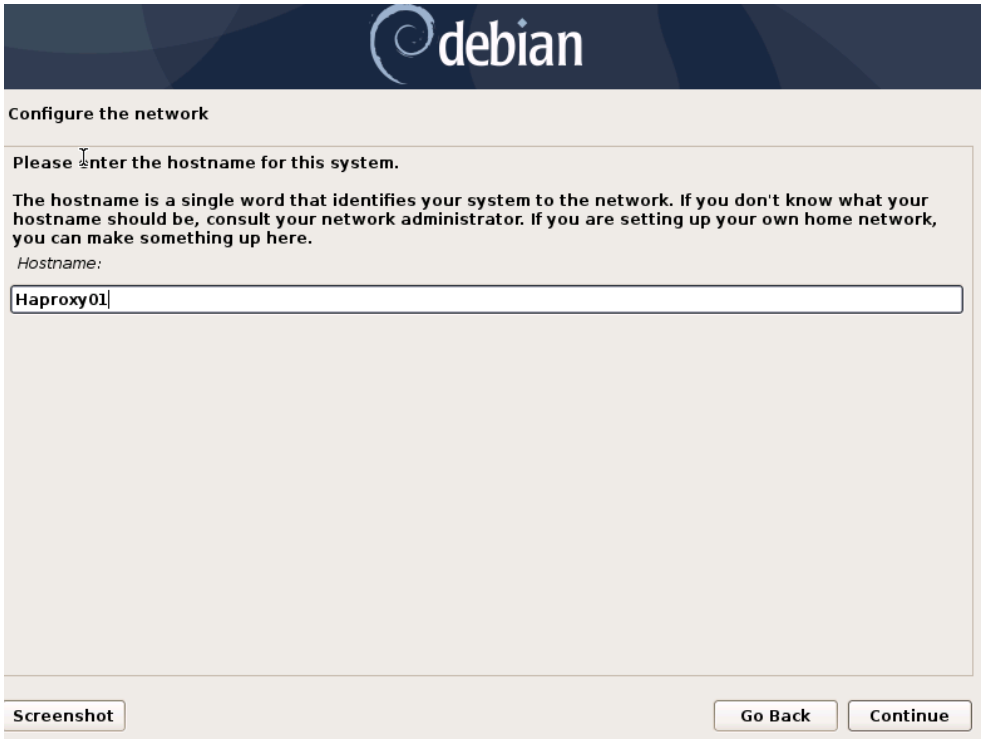
Paso 1

Ejecutamos el asistente de instalación:

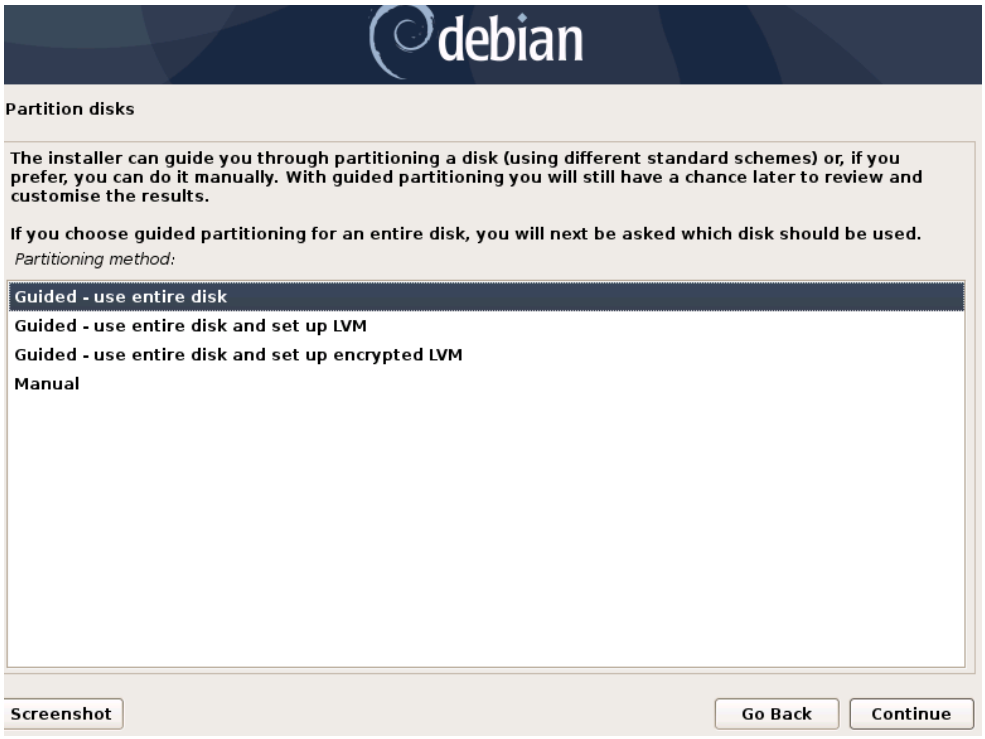
Seleccionaremos lenguaje de la instalación, localización, idioma teclado, etc...



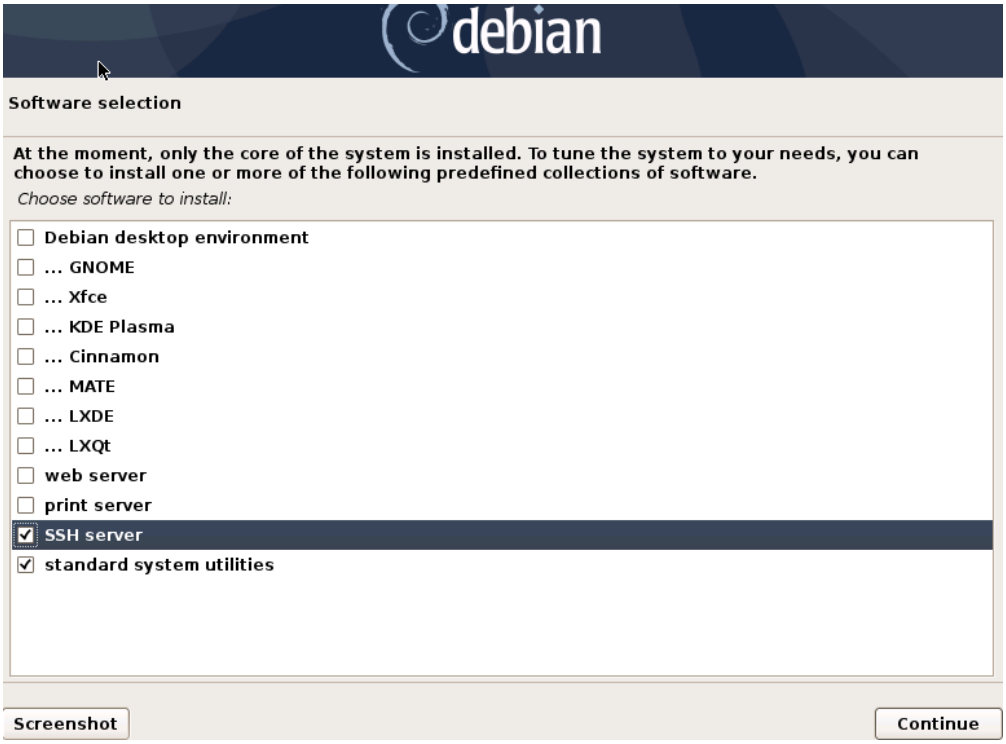
Indicaremos el nombre de host, dominio, usuarios y passwords.



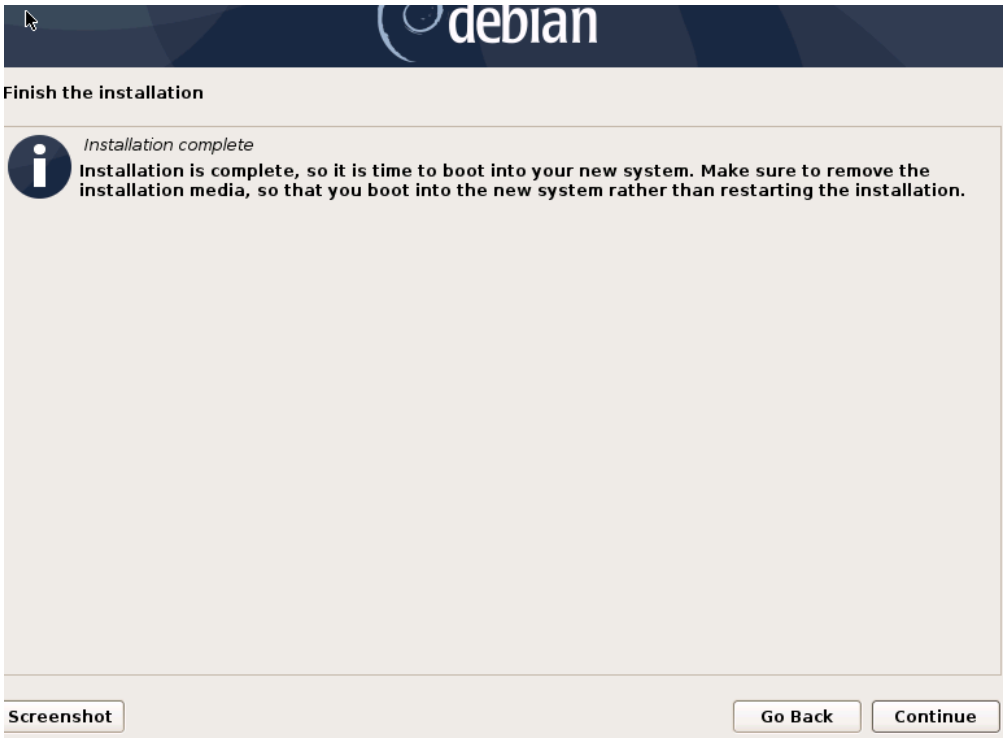
Realizamos el particionado de discos (usando la configuración por defecto). Indicamos una fuente de paquetes apt, e instalamos el sistema base.



No será necesario instalar un entorno de escritorio, pero sí instalaremos el servicio SSH

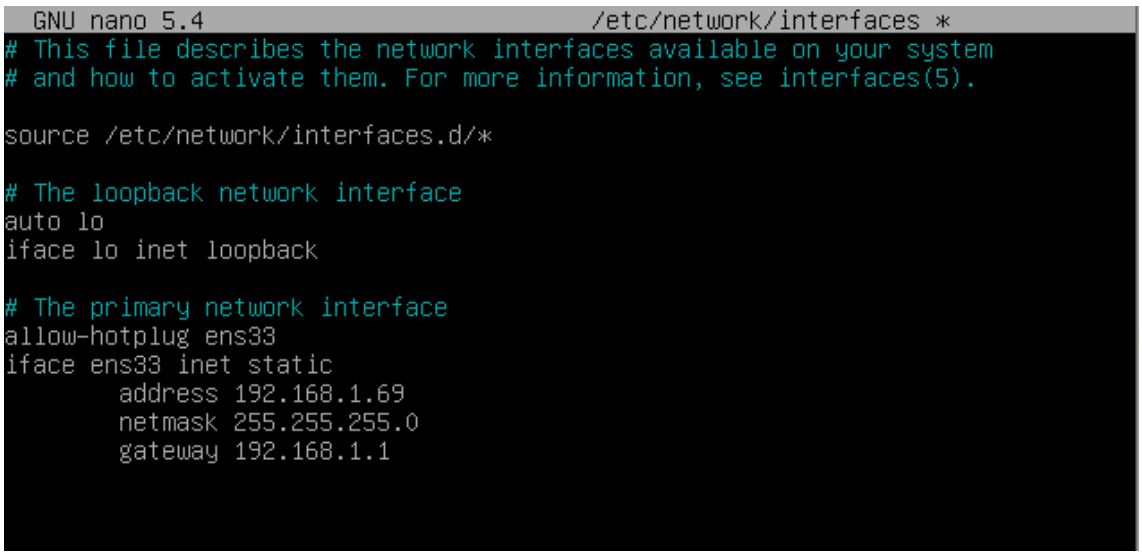


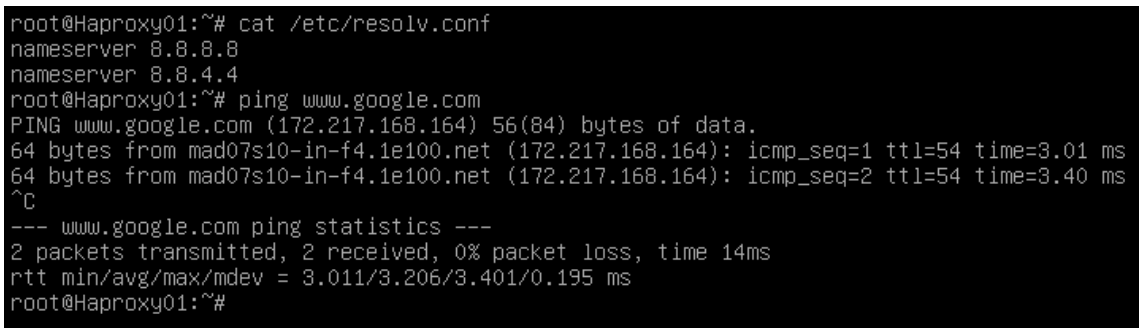
Finalizaremos la instalación del S.O.



Paso 2

Accedemos al servidor y configuramos los datos IP (si no lo hemos hecho durante la instalación del S.O.). Confirmamos que los servidores DNS son correctos y tenemos salida a internet:





Una vez configurados los datos IP, debemos ejecutar los comandos de actualización por si hubiera parches importantes se seguridad y de otros componentes que podamos aplicar:

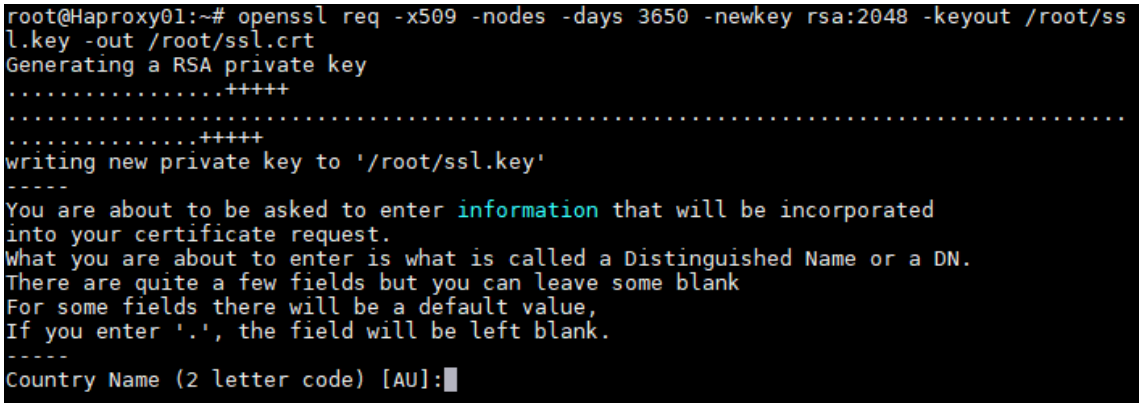
```
apt-get update
```

```
apt-get upgrade
```

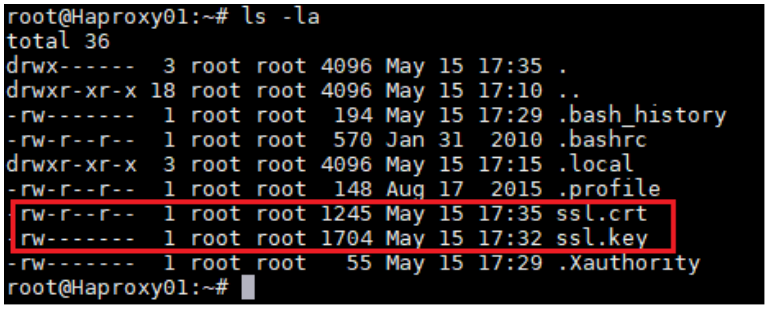
Paso 3

Si no disponemos de un certificado, generaremos uno temporal con el siguiente comando:

```
openssl req -x509 -nodes -days 3650 -newkey rsa:2048 -keyout /root/ssl.key -out /root/ssl.crt
```



Indicaremos todos los datos que nos solicite y confirmaremos que en la ruta especificada (/root) tenemos los ficheros ssl.key y ssl.crt

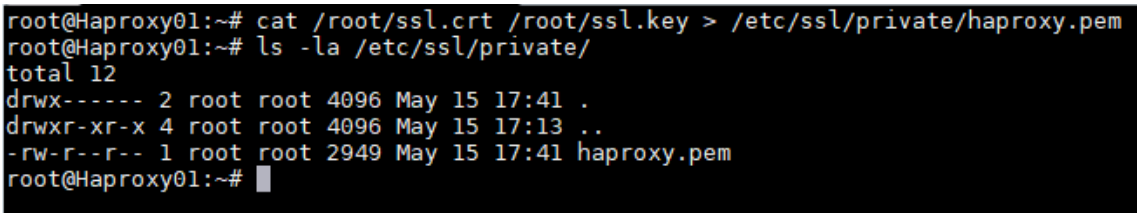


Ahora juntaremos ambos ficheros y crearemos el fichero .pem que será el que especifiquemos en la configuración del HAProxy.

Para crear el fichero. pem ejecutaremos el siguiente comando:

```
cat /root/ssl.crt /root/ssl.key > /etc/ssl/private/haproxy.pem
```

Creamos el nuevo fichero de certificado y confirmamos que está alojado en la ruta indicada:



NOTA:
Este certificado creado en el servidor HAProxy primario será necesario copiarlo a la misma ruta del servidor secundario.

Si se está utilizando un certificado propio, será necesario copiarlo en ambos servidores (primario y secundario).

Paso 4

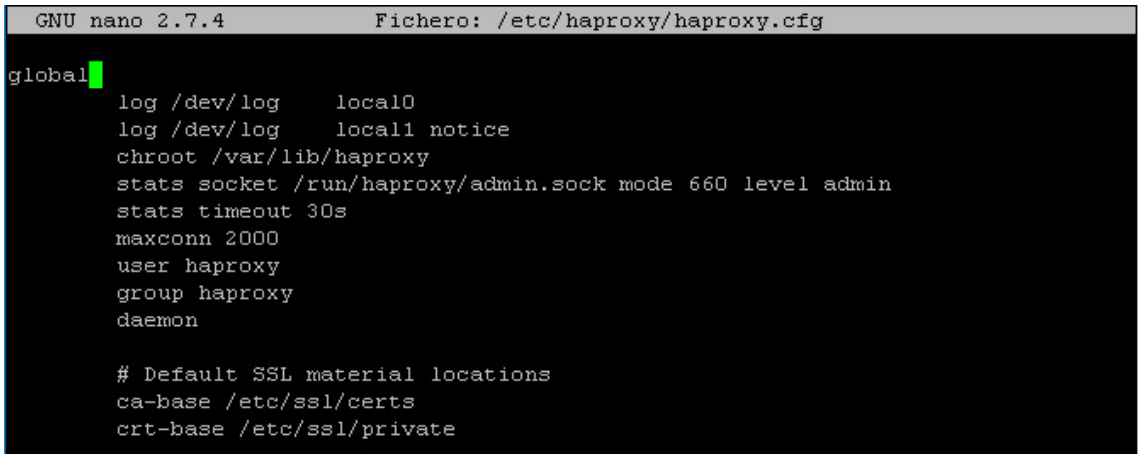
Realizamos la instalación del servicio HAProxy:

```
apt-get install haproxy
```

```
root@Haproxy01:~# apt-get install haproxy
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following additional packages will be installed:
  liblua5.3-0
Suggested packages:
  vim-haproxy haproxy-doc
The following NEW packages will be installed:
  haproxy liblua5.3-0
0 upgraded, 2 newly installed, 0 to remove and 0 not upgraded.
Need to get 1,424 kB of archives.
After this operation, 3,061 kB of additional disk space will be used.
Do you want to continue? [Y/n] y
Get:1 http://deb.debian.org/debian buster/main amd64 liblua5.3-0 amd64 5.3.3-1.1 [120 k
B]
Get:2 http://deb.debian.org/debian buster/main amd64 haproxy amd64 1.8.19-1+deb10u2 [1,
304 kB]
Fetched 1,424 kB in 1s (2,417 kB/s)
Selecting previously unselected package liblua5.3-0:amd64.
(Reading database ... 31798 files and directories currently installed.)
Preparing to unpack .../liblua5.3-0_5.3.3-1.1_amd64.deb ...
Unpacking liblua5.3-0:amd64 (5.3.3-1.1) ...
Selecting previously unselected package haproxy.
Preparing to unpack .../haproxy_1.8.19-1+deb10u2_amd64.deb ...
Unpacking haproxy (1.8.19-1+deb10u2) ...
Setting up liblua5.3-0:amd64 (5.3.3-1.1) ...
Setting up haproxy (1.8.19-1+deb10u2) ...
Created symlink /etc/systemd/system/multi-user.target.wants/haproxy.service → /lib/syst
emd/system/haproxy.service.
Processing triggers for man-db (2.8.5-2) ...
Processing triggers for libc-bin (2.28-10) ...
Processing triggers for rsyslog (8.1901.0-1) ...
Processing triggers for systemd (241-7~deb10u4) ...
root@Haproxy01:~#
```

Tras realizar la instalación del servicio HAProxy, editaremos el fichero de configuración **haproxy.cfg**, para configurar el servicio ubicado en la ruta /etc/haproxy/

Eliminaremos todo el contenido del fichero, añadiendo el siguiente texto (puede descargar el fichero del siguiente repositorio): https://images.udsenterprise.com/files/UDS_HA/HAProxy/3.6/haproxy.cfg




```
global
log /dev/log    local0
log /dev/log    local1 notice
chroot /var/lib/haproxy
stats socket /run/haproxy/admin.sock mode 660 level admin
stats timeout 30s
maxconn 2048
user haproxy
group haproxy
daemon

# Default SSL material locations
ca-base /etc/ssl/certs
crt-base /etc/ssl/private

# Default ciphers to use on SSL-enabled listening sockets.
# For more information, see ciphers(1SSL). This list is from:
#  https://hynek.me/articles/hardening-your-web-servers-ssl-ciphers/
ssl-default-bind-options ssl-min-ver TLSv1.2 prefer-client-ciphers
ssl-default-bind-ciphersuites
TLS_AES_128_GCM_SHA267:TLS_AES_267_GCM_SHA384:TLS_CHACHA20_POLY1305_SHA267
ssl-default-bind-ciphers ECDH+AESGCM:ECDH+CHACHA20:ECDH+AES267:ECDH+AES128:!aNULL:!SHA1:!AESCCM
```

```
# ssl-default-server-options ssl-min-ver TLSv1.2

# ssl-default-server-ciphersuites
TLS_AES_128_GCM_SHA267:TLS_AES_267_GCM_SHA384:TLS_CHACHA20_POLY1305_SHA267

#ssl-default-server-ciphers
ECDH+AESGCM:ECDH+CHACHA20:ECDH+AES267:ECDH+AES128:!aNULL:!SHA1:!AESCCM


tune.ssl.default-dh-param 2048




defaults
        log     global
        mode    http
        option  httplog
        option  dontlognull
        option  forwardfor
        retries 3
        option  redispatch

        stats enable
        stats uri /haproxystats
        stats realm Strictly\ Private
        stats auth stats:haproxystats

        timeout connect 5000
        timeout client  14000
        timeout server  14000
        errorfile 400 /etc/haproxy/errors/400.http
        errorfile 403 /etc/haproxy/errors/403.http
        errorfile 408 /etc/haproxy/errors/408.http
        errorfile 500 /etc/haproxy/errors/500.http
        errorfile 502 /etc/haproxy/errors/502.http
        errorfile 503 /etc/haproxy/errors/503.http
        errorfile 504 /etc/haproxy/errors/504.http
```

```
frontend https-in
        bind *:443 ssl crt /etc/ssl/private/haproxy.pem
        mode http
        http-request set-header X-Forwarded-Proto https
        default_backend uds-backend

frontend tunnel-in
        bind *:1443
        mode tcp
        option tcplog
        default_backend tunnel-backend-ssl

frontend tunnel-in-guacamole    # HTML5
        bind *:10443
        mode tcp
        option tcplog
        default_backend tunnel-backend-guacamole
backend uds-backend
        option http-keep-alive
        balance source
        server udss1 192.168.1.183:443 check inter 2000 rise 2 fall 5 ssl verify none
        server udss2 192.168.1.184:443 check inter 2000 rise 2 fall 5 ssl verify none
backend tunnel-backend-ssl
        mode tcp
        option tcplog
        balance roundrobin
        server udst1 192.168.1.185:443 check inter 2000 rise 2 fall 5
        server udst2 192.168.1.186:443 check inter 2000 rise 2 fall 5

backend tunnel-backend-guacamole
        mode tcp
        option tcplog
        balance source
        server udstg1 192.168.1.185:10443 check inter 2000 rise 2 fall 5
        server udstg2 192.168.1.186:10443 check inter 2000 rise 2 fall 5
```

```
# Galera Cluster Frontend configuration
frontend galera_cluster_frontend
        bind *:3306
        mode tcp
        option tcplog
        default_backend galera_cluster_backend

# Galera Cluster Backend configuration
backend galera_cluster_backend
        mode tcp
        option tcpka
        balance source
        server dbserver1 192.168.1.180:3306 check weight 1
        server dbserver2 192.168.1.181:3306 check weight 2
        server dbserver3 192.168.1.182:3306 check weight 3
```

Dónde:

Ruta de los certificados.

```
# Default SSL material locations
ca-base /etc/ssl/certs
crt-base /etc/ssl/private
```

Acceso a las estadísticas.

```
            stats enable
            stats uri /haproxystats
            stats realm Strictly\ Private
            stats auth stats:haproxystats
```

Regla acceso Frontend al servidor UDS en modo http (indicaremos la ruta del certificado .pem generado anteriormente). Puerto 443.

```
frontend https-in
        bind *:443 ssl crt /etc/ssl/private/haproxy.pem
        mode http
        http-request set-header X-Forwarded-Proto https
        default_backend uds-backend
```

Regla acceso Frontend al servidor Tunnel en modo TCP por el **puerto 1443** (conexiones tunelizadas). En caso de utilizar otro puerto diferente, será necesario modificarlo (este puerto es el que ha sido indicado en la pestaña Tunnel de un transporte vía tunnel).

```
frontend tunnel-in
         bind *:1443
         mode tcp
         option tcplog
         default_backend tunnel-backend-ssl
```

Regla acceso Frontend al servidor Tunnel en modo TCP por el **puerto 10443** (conexiones HTML5). En caso de utilizar otro puerto diferente será necesario modificarlo (este puerto es el que ha sido indicado en la pestaña tunnel de un transporte HTML5).

```
frontend tunnel-in-guacamole    # HTML5
        bind *:10443
        mode tcp
        option tcplog
        default_backend tunnel-backend-guacamole
```

Regla de acceso Backend al servidor UDS. **Deberemos indicar las direcciones IP de nuestras máquinas UDS-Server** (el puerto de escucha del servidor UDS es el 443).

```
backend uds-backend
        option http-keep-alive
        balance source
        server udss1 192.168.1.183:443 check inter 2000 rise 2 fall 5
        server udss2 192.168.1.184:443 check inter 2000 rise 2 fall 5
```

Regla de acceso backend al servidor Tunnel para las conexiones tunelizadas. **Deberemos indicar las direcciones IP de nuestras máquinas UDS-Tunnel** (el puerto de escucha del servidor Tunnel para las conexiones tunelizadas es 443).

```
backend tunnel-backend-ssl
        mode tcp
        option tcplog
        balance roundrobin
        server udst1 192.168.1.185:443 check inter 2000 rise 2 fall 5 ssl verify none
        server udst2 192.168.1.186:443 check inter 2000 rise 2 fall 5 ssl verify none
```

Regla de acceso backend al servidor Tunnel para las conexiones HTML5. **Deberemos indicar las direcciones IP de nuestras máquinas UDS-Tunnel** (el puerto de escucha del servidor Tunnel para las conexiones HTML5 es 10443).

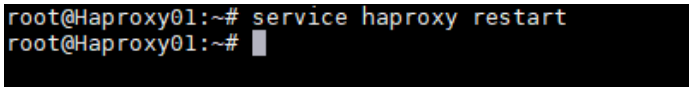
```
backend tunnel-backend-guacamole
        mode tcp
        option tcplog
        balance source
        server udstg1 192.168.1.185:10443 check inter 2000 rise 2 fall 5
        server udstg2 192.168.1.186:10443 check inter 2000 rise 2 fall 5
```

Reglas de acceso a las diferentes base de datos, deberemos indicar las diferentes ips de nuestros servidores de base de datos (el puerto de escucha para las conexiones con las bases de datos es 3306)

```
        bind *:3306
        mode tcp
        option tcplog
        default_backend galera_cluster_backend


backend galera_cluster_backend
        mode tcp
        option tcpka
        balance source
        server dbserver1 192.168.1.67:3306 check weight 30
        server dbserver2 192.168.1.69:3306 check weight 20
        server dbserver3 192.168.1.70:3306 check weight 10
```

Tras realizar la configuración del fichero, lo guardamos y reiniciamos el servicio HAProxy:

```
service haproxy restart
```



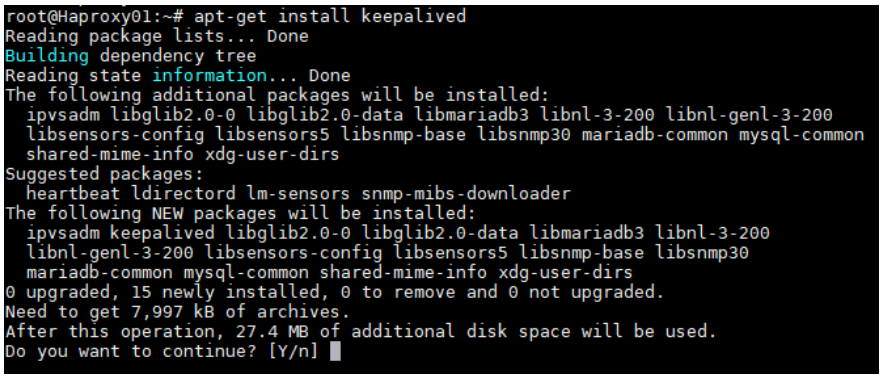
Paso 5

Una vez que hemos terminado la instalación y configuración de HAProxy, instalaremos keepalive, el cual nos proporcionará una ip virtual de balanceo entre los diferentes servidores HAProxy.

Ante una caída del servidor principal HAProxy, la IP virtual de balanceo se activará automáticamente en el servidor secundario. Una vez recuperado el servicio en el servidor principal, la IP virtual volverá a activarse en dicho servidor.

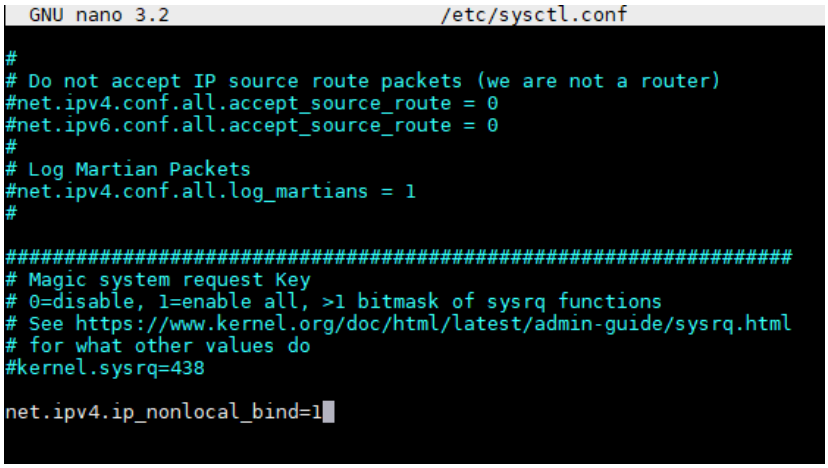
Para realizar la instalación de Keepalive, ejecutaremos el siguiente comando:

```
apt-get install keepalived
```



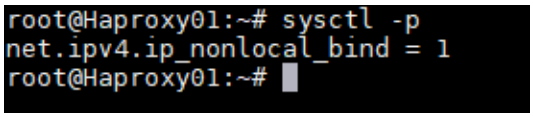
Una vez instalado, editaremos el fichero /etc/sysctl.conf y añadiremos la siguiente línea al final del fichero:

```
net.ipv4.ip_nonlocal_bind=1
```



Para verificar que la modificación se ha realizado correctamente, podremos ejecutar el siguiente comando:

```
sysctl -p
```



Ahora configuraremos el servicio Keepalived. Para ello creamos el fichero keepalived.conf en la ruta /etc/keepalived/

Depende del nodo que estemos configurando (principal o secundario), tendremos que indicar una configuración:

- **FICHERO KEEPALIVED.CONF EN NODO PRINCIPAL**

El fichero se puede descargar del siguiente repositorio:

https://images.udsenterprise.com/files/UDS_HA/HAProxy/3.6/keepalived-master/keepalived.conf

En caso de crearlo manualmente, deberemos indicar lo siguiente:

```
global_defs {
# Keepalived process identifier
lvs_id haproxy_DH
}
# Script used to check if HAProxy is running
vrrp_script check_haproxy {
script "killall -0 haproxy"
interval 2
weight 2
}
# Virtual interface
# The priority specifies the order in which the assigned interface to take over in a
failover
vrrp_instance VI_01 {
state MASTER
interface enp1s0
virtual_router_id 51
priority 101
# The virtual ip address shared between the two loadbalancers
virtual_ipaddress {
192.168.11.64/24
}
track_script {
check_haproxy
}
}
```

Dónde:

Indicaremos el nombre de la interfaz de red de la máquina (con el comando `ip a` podremos comprobar el nombre de nuestro interfaz de red):

```
interface enp1s0
```

Definiremos el rol del servidor (MASTER= principal, SLAVE= secundario)

```
state MASTER
```

Indicaremos la dirección IP virtual de balanceo:

```
virtual_ipaddress {
192.168.1.189/24
```

```
}
```

```
  GNU nano 5.4                                  /etc/keepalived/keepalived.conf *
global_defs {
# Keepalived process identifier
lvs_id haproxy_DH
}
# Script used to check if HAProxy is running
vrrp_script check_haproxy {
script "killall -0 haproxy"
interval 2
weight 2
}
# Virtual interface
# The priority specifies the order in which the assigned interface to take over in a failover
vrrp_instance VI_01 {
state MASTER
interface enp1s0
virtual_router_id 51
priority 101
# The virtual ip address shared between the two loadbalancers
virtual_ipaddress {
192.168.1.189
}
track_script {
check_haproxy
}
}
```

- **FICHERO KEEPALIVED.CONF EN NODO SECUNDARIO**

El fichero se puede descargar del siguiente repositorio:

https://images.udsenterprise.com/files/UDS_HA/HAProxy/3.6/keepalived-slave/keepalived.conf

En caso de crearlo manualmente, deberemos indicar lo siguiente:

```
global_defs {
# Keepalived process identifier
lvs_id haproxy_DH_passive
}
# Script used to check if HAProxy is running
vrrp_script check_haproxy {
script "killall -0 haproxy"
interval 2
weight 2
}
# Virtual interface
# The priority specifies the order in which the assigned interface to take over in a
failover
vrrp_instance VI_01 {
state SLAVE
interface enp1s0
virtual_router_id 51
priority 100
# The virtual ip address shared between the two loadbalancers
virtual_ipaddress {
192.168.1.189/24
}
track_script {
check_haproxy
}
}
```

Dónde:

Indicaremos el nombre de la interfaz de red de la máquina (con el comando `ip a` podremos comprobar el nombre de nuestro interfaz de red):

```
interface enp1s0
```

Definiremos el rol del servidor (MASTER= principal, SLAVE= secundario)

```
state SLAVE
```

Indicaremos la dirección IP virtual de balanceo

```
virtual_ipaddress {
192.168.1.189/24
}
```

```
  GNU nano 5.4                                  /etc/keepalived/keepalived.conf *
global_defs {
# Keepalived process identifier
lvs_id haproxy_DH_passive
}
# Script used to check if HAProxy is running
vrrp_script check_haproxy {
script "killall -0 haproxy"
interval 2
weight 2
}
# Virtual interface
# The priority specifies the order in which the assigned interface to take over in a failover
vrrp_instance VI_01 {
state SLAVE
interface enp1s0
virtual_router_id 51
priority 100
# The virtual ip address shared between the two loadbalancers
virtual_ipaddress {
192.168.1.189
}
track_script {
check_haproxy
}
}
```

Una vez creados los ficheros en ambos servidores (principal y secundario), será necesario reiniciar el servicio keepalived:

```
service keepalived restart
```

```
root@Haproxy01:~# service keepalived restart
root@Haproxy01:~#
```

Verificamos con el comando `ip a` que la IP virtual de balanceo está activa en el servidor principal:

```
root@haproxy1:~# 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: enp1s0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP group default qlen 1000
    link/ether 52:54:00:99:e0:c5 brd ff:ff:ff:ff:ff:ff
    inet 192.168.1.187/20 brd 192.168.15.255 scope global enp1s0
       valid_lft forever preferred_lft forever
    inet 192.168.1.189/32 scope global enp1s0
       valid_lft forever preferred_lft forever
    inet6 fe80::5054:ff:fe99:e0c5/64 scope link
       valid_lft forever preferred_lft forever
root@haproxy1:~# _
```

Configuración de los servidores UDS Server y Tunnel

Una vez configurados los servidores de base de datos y los servidores HAProxy a modo de balanceadores, procederemos a instalar y configurar los componentes UDS-Server y UDS-Tunnel.

Comenzaremos por el componente UDS-Server, puesto que la configuración de las máquinas UDS-Tunnel nos requerirá tener al menos una máquina UDS-Server activa y configurada.

Configuración servidores UDS (UDS-Server)

Iniciaremos las máquinas UDS-Server y procederemos a su configuración.

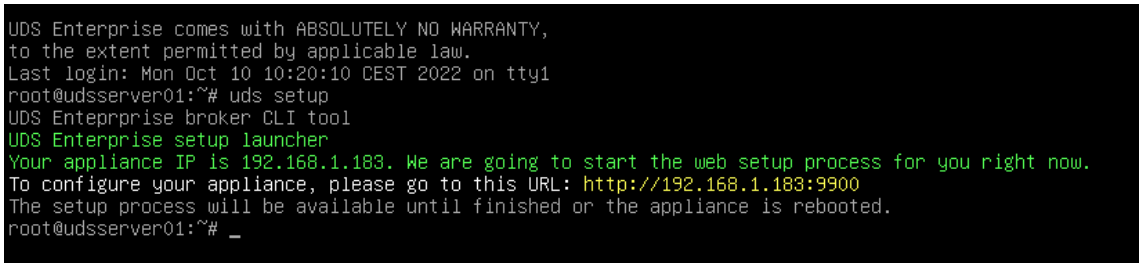
La primera tarea será asignar una dirección IP al servidor para poder acceder al asistente de configuración vía navegador. Para ello ejecutaremos el comando:

`uds ip set` *`dirección_IP/mascara gateway hostname`*

```
root@uds:~# uds ip set 192.168.1.183/255.255.255.0 192.168.1.1 udsserver01
UDS Enteprprise broker CLI tool
Updating network configuration...[  104.143036] e1000: eth0 NIC Link is Up 1000 Mbps Full Duplex, Fl
ow Control: None
done
New network configuration
DHCP: no
Using interface: eth0
Hostname: udsserver01
Address: 192.168.1.183
Mask: 255.255.255.0
Gateway: 192.168.1.1
DNS: 80.58.61.250
Secondary DNS: 80.58.61.254
You need to reboot your appliance in order to fully activate the new configuration
root@uds:~#
```

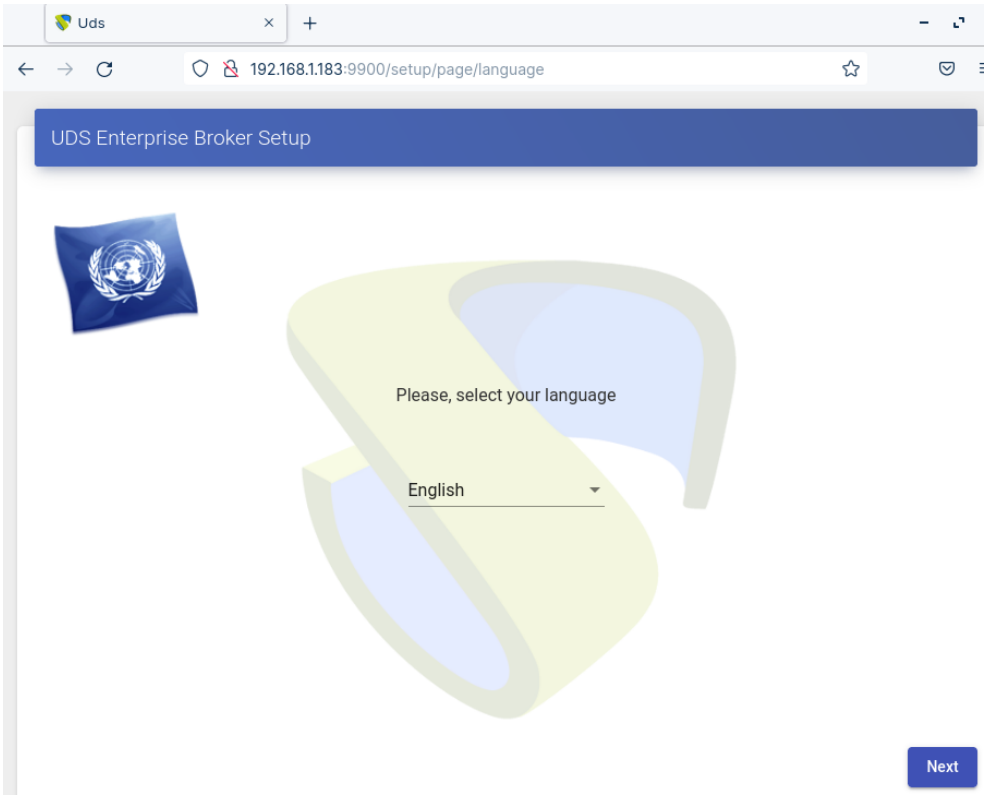
Después de indicar los datos IP, reiniciamos el servidor para aplicar los cambios

Si la red donde hemos desplegado el servidor UDS dispone de un servidor DHCP, este tomará una dirección IP vía DHCP que nos servirá para acceder al asistente de configuración:



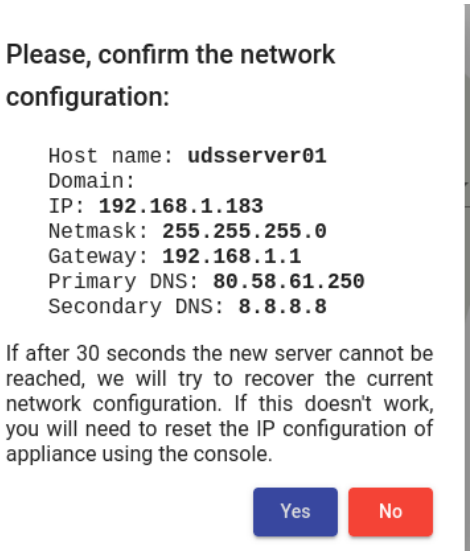
A través de un navegador, accedemos a la URL indicada para iniciar el asistente de configuración del servidor UDS (en este ejemplo: https://192.168.1.183:9900).

Seleccionamos el idioma del asistente de configuración:

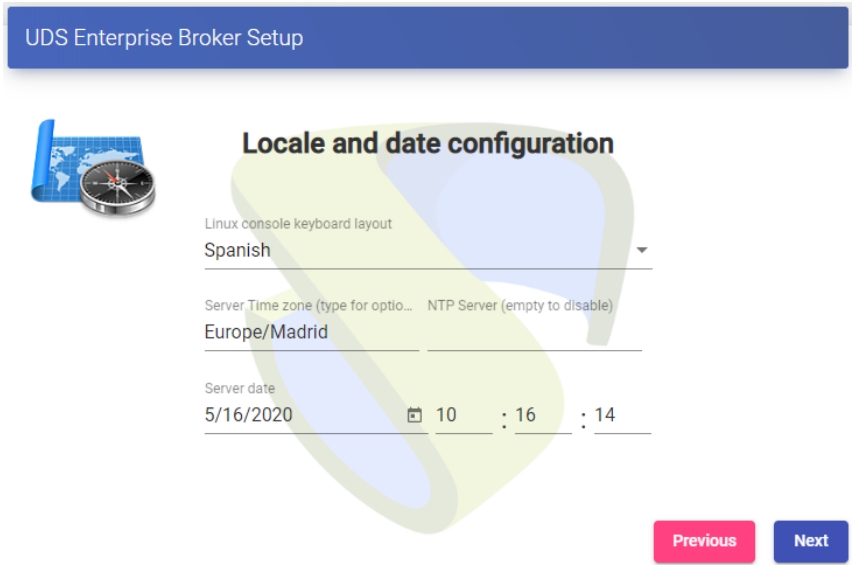


En el apartado de redes, indicamos los datos IP, nombre y dominio (opcional) que tendrá nuestro servidor UDS:

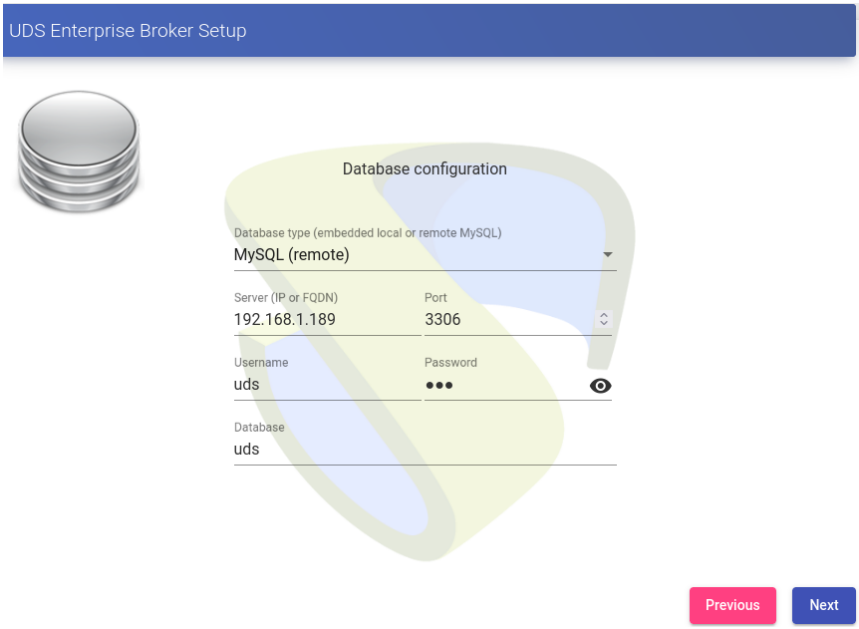
Confirmamos que los datos son correctos. Se procederá a aplicar los nuevos datos (en caso de acceder vía una dirección DHCP e indicar una dirección diferente, automáticamente se nos redirigirá, en el navegador, a la nueva dirección IP).



Seleccionamos el idioma del teclado, la zona horaria y opcionalmente podremos indicar un servidor NTP



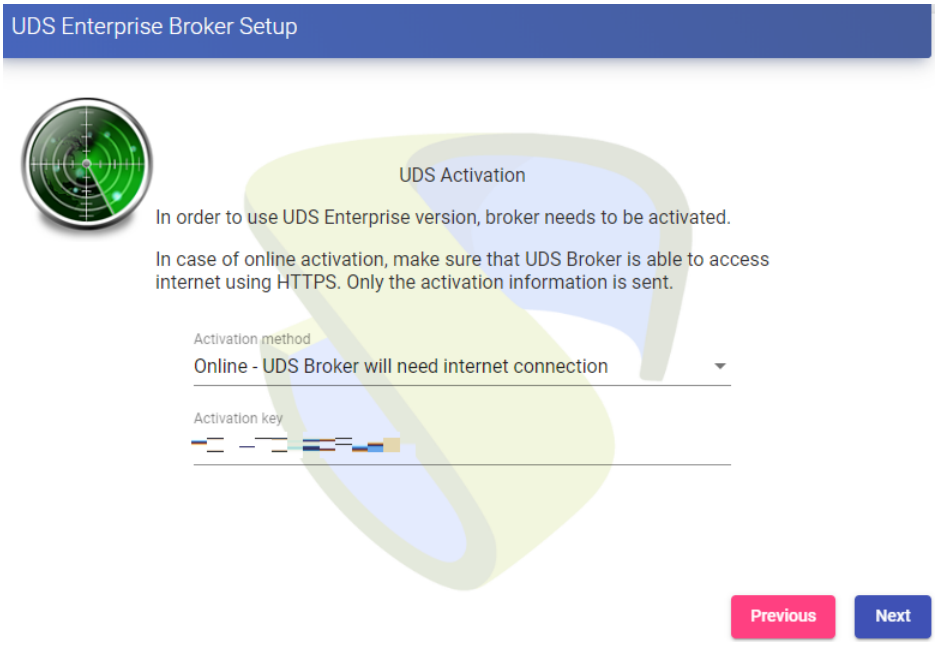
Ahora seleccionamos el tipo de base de datos: MySQL (remote) indicando los datos del servidor **MySQL principal**



En este caso tendremos que usar la ip virtual que comparten los HAProxy

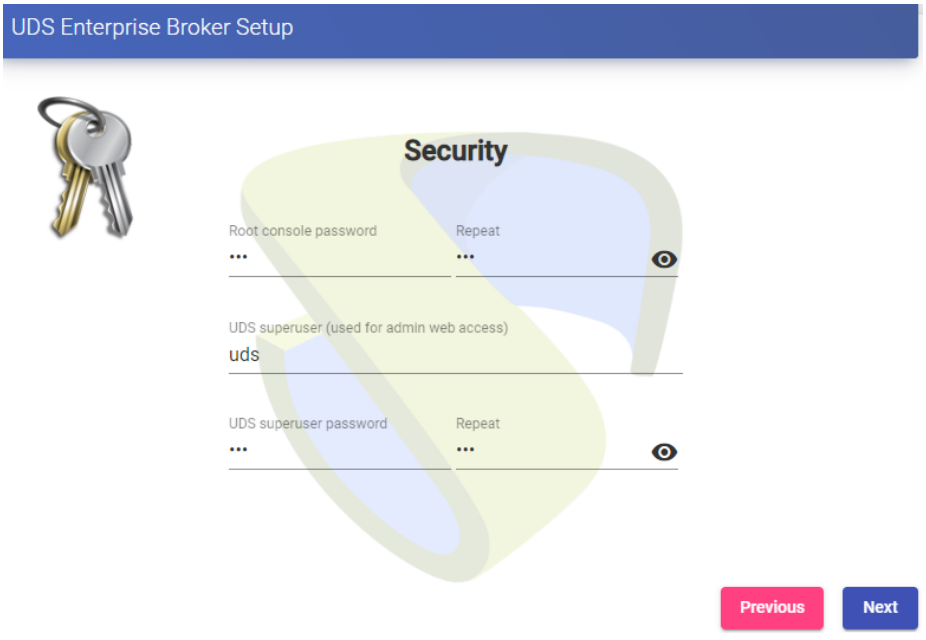
(192.168.1.189 en este caso)

La siguiente tarea será la de activar nuestro servidor UDS con un número de serie válido. En este ejemplo utilizaremos el método de activación online, el cual requiere que la máquina UDS-Server disponga de salida a internet.

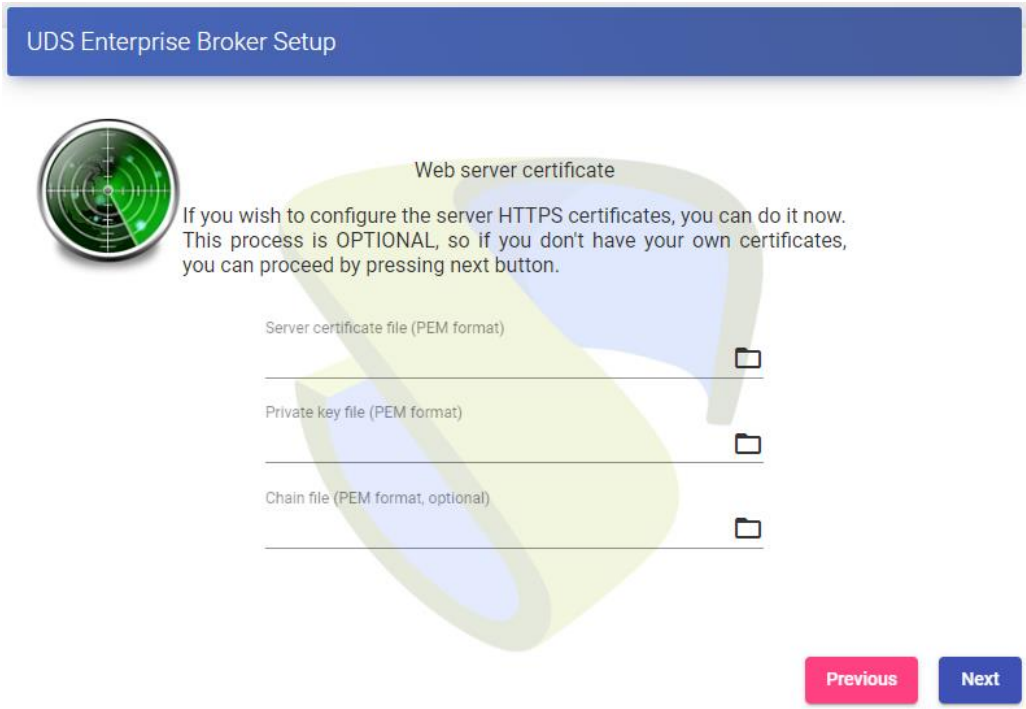


NOTA:
Si los servidores UDS no disponen de salida a internet, deberemos aplicar el proceso de activación offline (para más información de este procedimiento, puede consultar el Manual de Instalación, Administración y Usuario de UDS Enterprise disponible en la sección de **Documentación** de la página web udsenterprise.com)

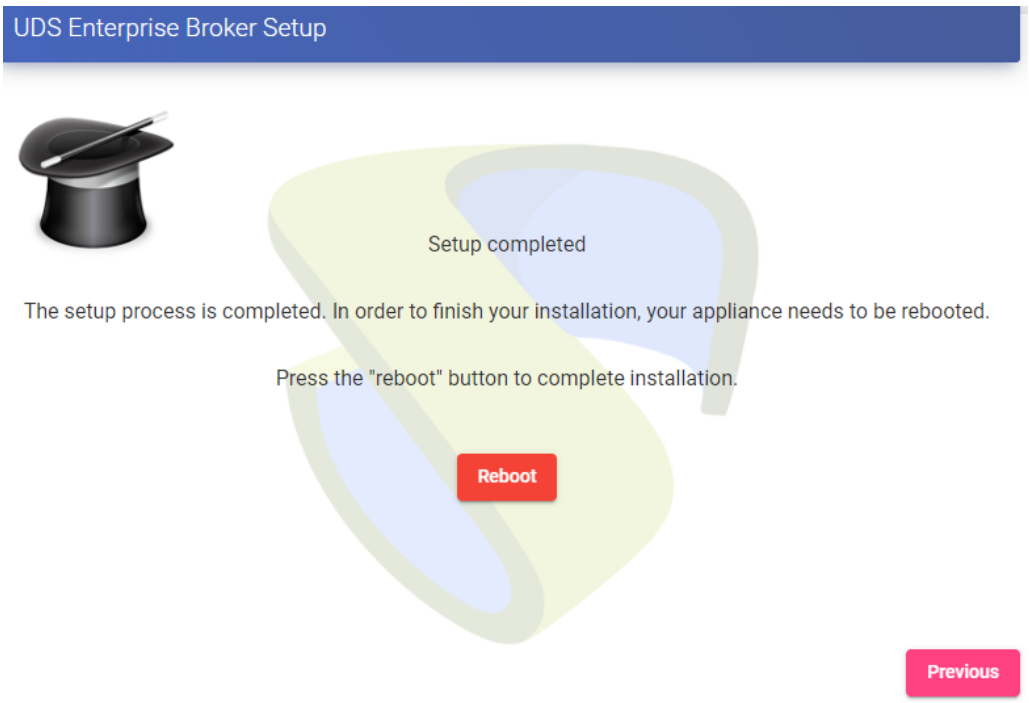
Indicaremos las credenciales del superusuario, el cual tendrá acceso a la administración de UDS. La contraseña indicada también será aplicada al usuario root del S.O. Linux que aloja el servicio de UDS:



Podremos instalar los certificados en el servidor UDS. En este caso al acceder vía balanceador (HAProxy), no será necesario instalarlos, aunque si se desea que la comunicación entre los componentes UDS-Server y UDS-Tunnel se realice vía HTTPS, sí será necesaria su configuración.

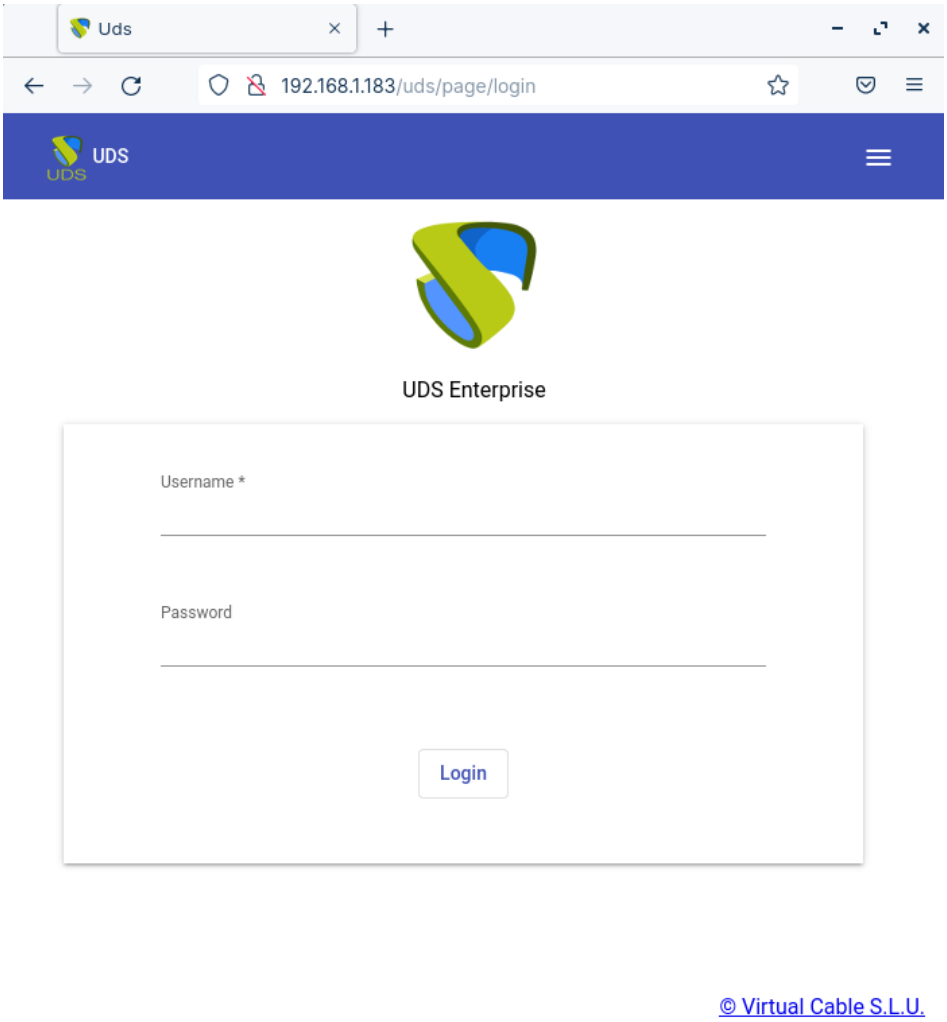


Reiniciaremos el servidor para finalizar su proceso de configuración.



Una vez reiniciado el servidor, ya podremos acceder al entorno UDS. El acceso lo realizaremos vía nombre o dirección IP de los datos configurados en la dirección IP virtual de balanceo configurada en el servidor HAProxy.

El primer acceso lo realizaremos con el superusuario configurado en el asistente de configuración:



Deberemos repetir todos los pasos anteriormente detallados en la segunda máquina UDS-Server. Lógicamente, los datos IP y nombre del segundo servidor serán diferentes, pero sí debemos conectar con la misma instancia de base de datos (nodo principal) e indicar el mismo número de serie para la activación.

Ambos servidores funcionarán en modo activo/activo y en caso de caída de uno de ellos, todas las peticiones de login se realizarán sobre el nodo activo de forma automática.

Configuración servidores Tunnel (UDS-Tunnel)

Iniciaremos las máquinas UDS-Tunnel y procederemos a su configuración.

La primera tarea será asignar una dirección IP al servidor para poder acceder al asistente de configuración vía navegador. Para ello ejecutaremos el comando:

`uds ip set` *`dirección_IP/mascara gateway hostname`*

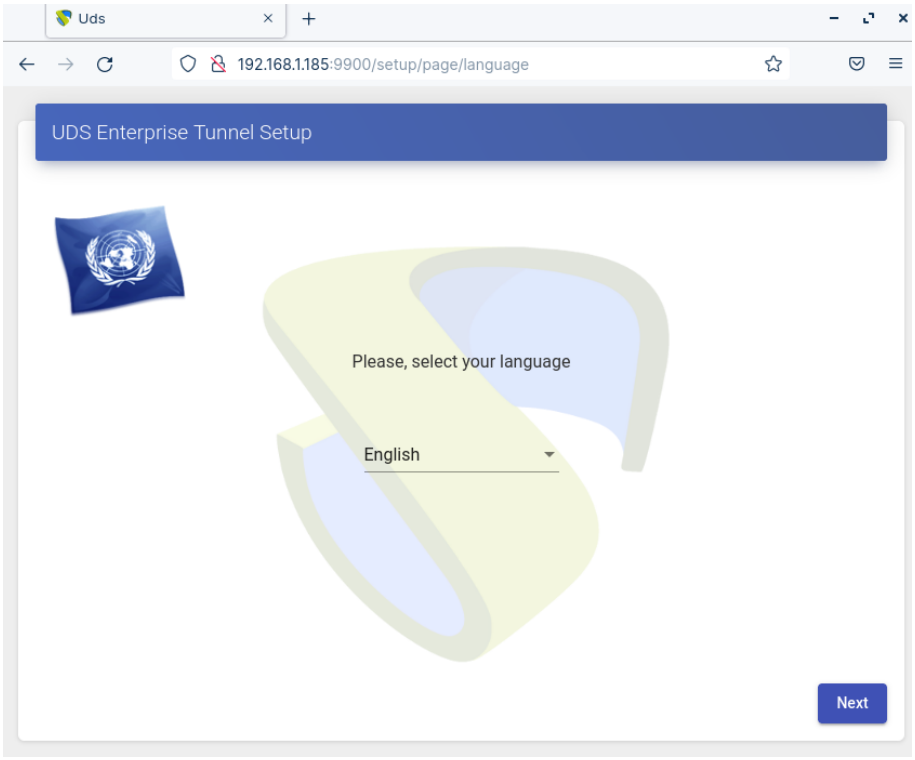
```
root@tunnel:~# uds ip set 192.168.1.185/255.255.255.0 192.168.1.1 udstunel
UDS Enteprprise tunnel CLI tool
Updating network configuration...[   62.746389] e1000: eth0 NIC Link is Up 1000 Mbps Full Duplex
ow Control: None
done
New network configuration
DHCP: no
Using interface: eth0
Hostname: udstunel
Address: 192.168.1.185
Mask: 255.255.255.0
Gateway: 192.168.1.1
DNS: 80.58.61.250
Secondary DNS: 80.58.61.254
You need to reboot your appliance in order to fully activate the new configuration
root@tunnel:~# _
```

Después de indicar los datos IP, reiniciamos el servidor para aplicar los cambios.

Si la red donde hemos desplegado el servidor Tunnel dispone de un servidor DHCP, este tomará una dirección IP vía DHCP que nos servirá para acceder al asistente de configuración.

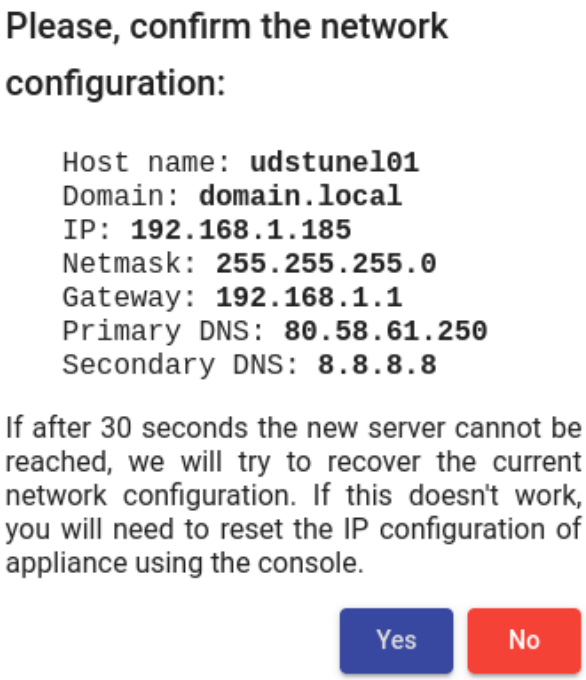
A través de un navegador, accedemos a la URL indicada para iniciar el asistente de configuración del servidor Tunnel (en este ejemplo: https://192.168.1.185:9900).

Seleccionamos el idioma del asistente de configuración:

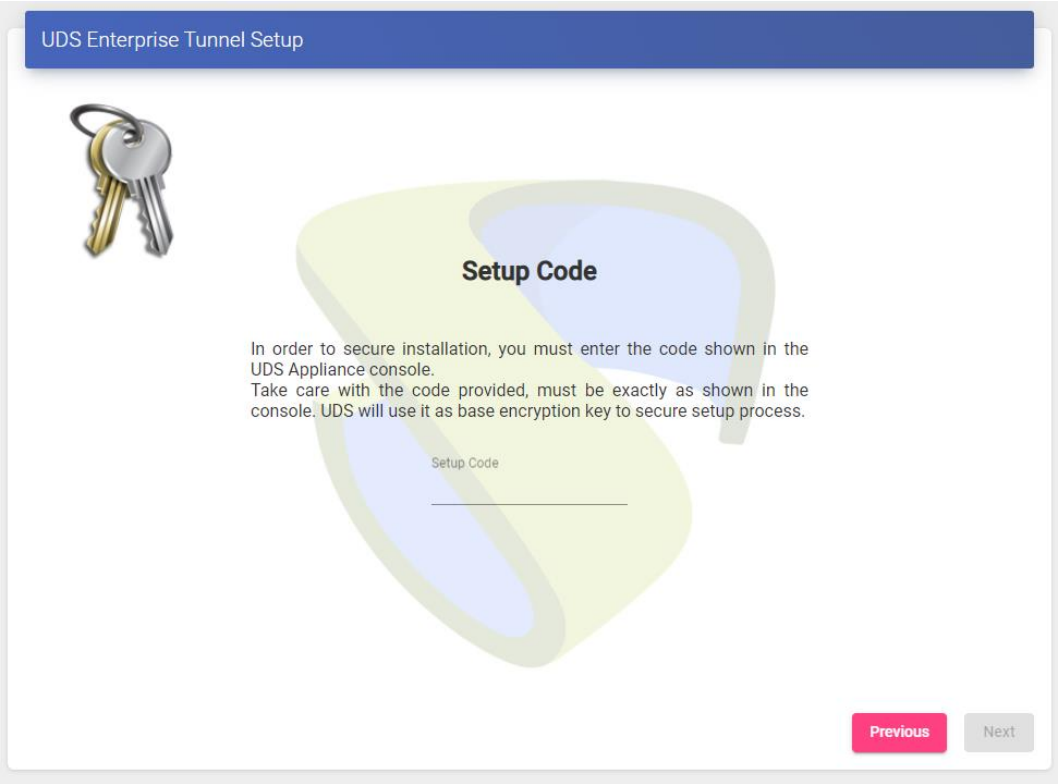


En el apartado de redes, indicamos los datos IP, nombre y dominio (opcional) que tendrá nuestro servidor Tunnel:

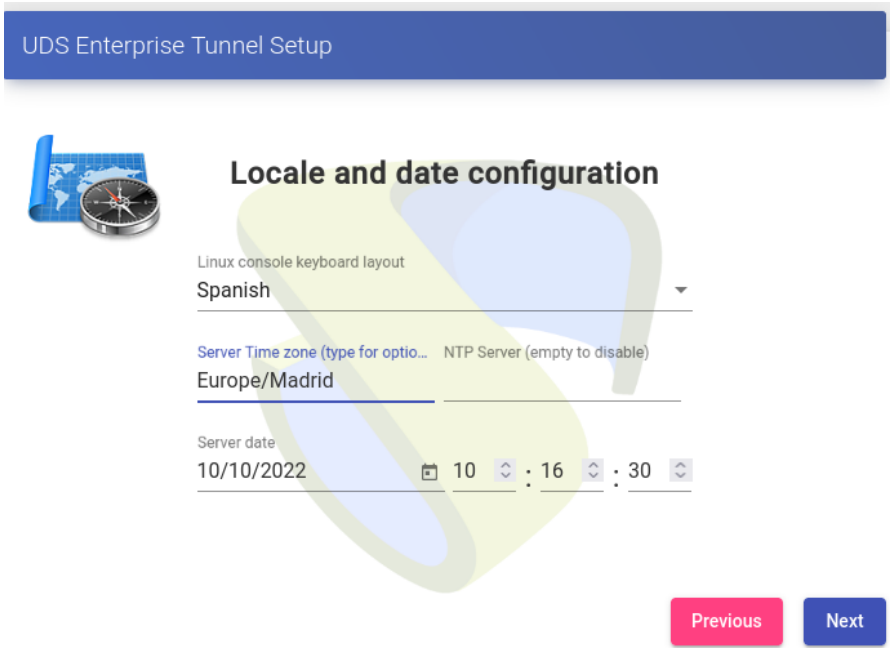
Confirmamos que los datos son correctos. Se procederá a aplicar los nuevos datos (en caso de acceder vía una dirección DHCP e indicar una dirección diferente, automáticamente se nos redirigirá, en el navegador, a la nueva dirección IP).



A continuación, añadiremos el código de seguridad que aparece en nuestro Appliance:



Seleccionamos el idioma del teclado, la zona horaria y opcionalmente podremos indicar un servidor NTP:



Seleccionamos cómo se realizará la conexión con el servidor UDS e indicamos su dirección IP. Como en este caso está configurado a través de un balanceador (HAProxy), dicha dirección será la IP virtual de balanceo configurada anteriormente en el servidor HAProxy usando el servicio Keepalived.

Para que el UDS Tunnel confíe en el certificado autofirmado del HAProxy y poder validar la conexión tendremos que usar el comando "uds trust"

```
root@tunnel-360:~# uds trust -h
UDS Enteprprise tunnel CLI tool
usage: uds trust [-h] [-c] HOSTNAME PORT

positional arguments:
  HOSTNAME     Hostname of the remote server.
  PORT         Port of the remote server.

optional arguments:
  -h, --help   show this help message and exit
  -c, --chain  Trust the certificate full chain.
root@tunnel-360:~#
```

```
root@tunnel-360:~# uds trust 192.168.1.189 443
UDS Enteprprise tunnel CLI tool
Reading certificate from server 192.168.1.189:443 done
Certificate name: vc
Valid from: 2020-04-28 20:09:47
Valid until: 2030-04-26 20:09:47
Fingerprint: d0d77e63553c2fc00583a3763670a4b5732a320bbe4c994fc331a194bcc72393
Issuer: CN=vc,O=vc,ST=madrid,C=es
Subject: CN=vc,O=vc,ST=madrid,C=es
Serial number: 705584093455247764462372607821631288138207695058
Self signed: Yes
Writing certificate to trust file (/usr/local/share/ca-certificates/vc.crt)... done
Ensuring that the name vc resolves to the IP 192.168.1.189...
updating /etc/hosts... done
Updating trusted database...
Updating certificates in /etc/ssl/certs...
1 added, 0 removed; done.
Running hooks in /etc/ca-certificates/update.d...

Adding debian:vc.pem
done.
done.
Trusted certificate installed
root@tunnel-360:~# _
```

Una vez realizado tendremos que indicarle al UDS Tunnel el nombre de nuestro UDS Server “vc”

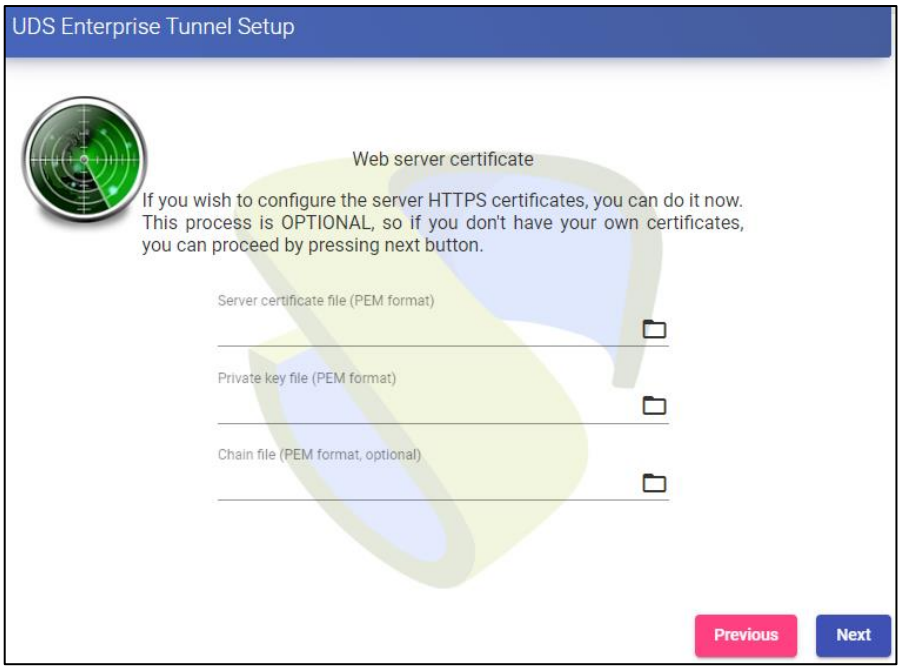
Editando el archivo /etc/hosts

```
  GNU nano 5.4                                    /etc/hosts *
# Autogenerated by UDS installer
127.0.0.1       localhost
127.0.1.1       tunnel-360.domain.local tunnel-360

# The following lines are desirable for IPv6 capable hosts
::1     ip6-localhost ip6-loopback
fe00::0 ip6-localnet
ff00::0 ip6-mcastprefix
ff02::1 ip6-allnodes
ff02::2 ip6-allrouters
192.168.1.189   vc
```

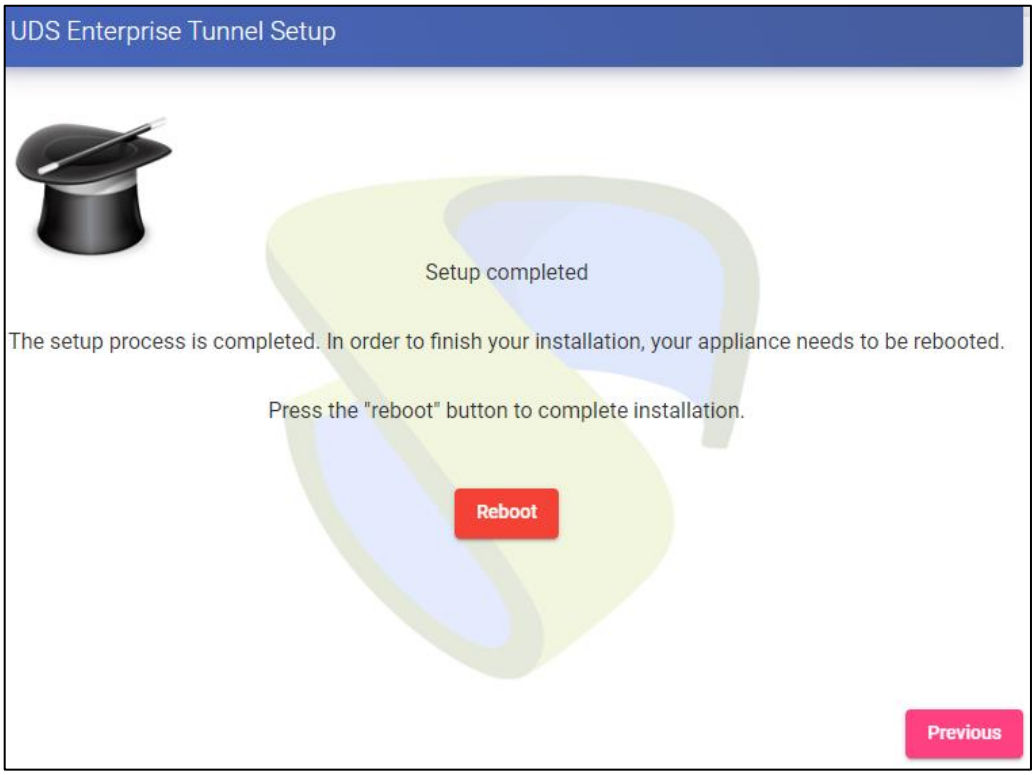
Una vez realizado el proceso podremos continuar con la configuración del Tunnel

NOTA:
Para más información sobre el comando uds ip, consultar el manual de Instalación, administración y usuario de UDS Enterprise 3.6

Podremos instalar los certificados en el servidor Tunnel para que las conexiones HTML5 dispongan de un certificado válido (en este ejemplo de dejaran los certificados autofirmados por defecto).



Reiniciaremos el servidor para finalizar su proceso de configuración.



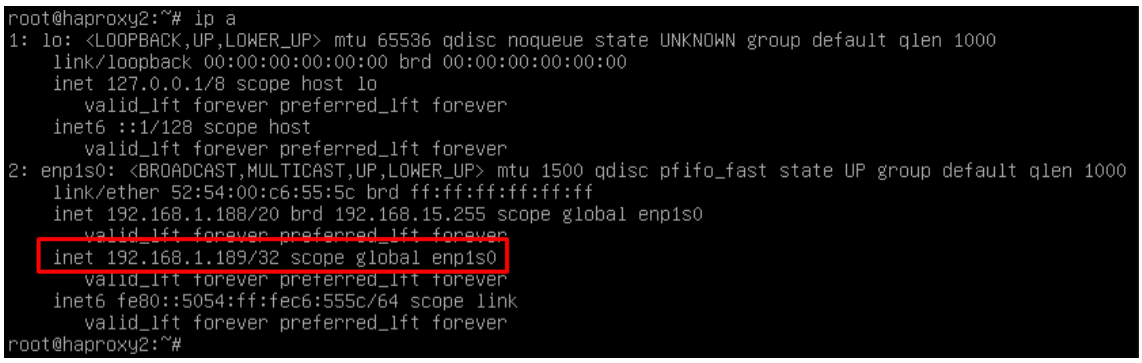
Una vez reiniciado el servidor, ya estará disponible para ser utilizado en conexiones tunelizadas (RDP, X2Go, Spice, etc...) y HTML5.

Deberemos repetir todos los pasos anteriormente detallados en la segunda máquina UDS-Tunnel. Lógicamente los datos IP y nombre del segundo servidor serán diferentes, pero sí debemos conectar con la misma dirección IP virtual de balanceo para proporcionar acceso conexión con los servidores UDS.

Ambos servidores funcionarán en modo activo/activo, cada usuario que realice una conexión vía tunnel se conectarán de forma aleatoria a estos servidores. En caso de caída de uno de ellos, las conexiones de los usuarios que estén usando ese servidor se cortará, pero al volver a realizar dicha conexión accederá a través del servidor Tunnel activo de forma automática.

Respuesta ante caída del servidor HA Proxy Maestro

En el caso de caída del servidor de HA Proxy maestro automáticamente la ip virtual que tiene nuestro servidor esclavo pasará a estar activa:



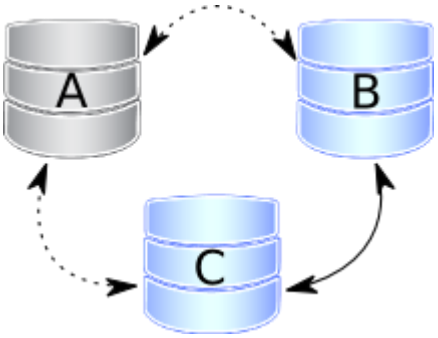
Con este cambio que se realiza automáticamente se podrá seguir trabajando con total normalidad, en el caso de levantamiento del servidor maestro, la ip virtual volverá a activarse en el servidor maestro.

Tareas de recuperación del clúster de Galera

Un clúster de Galera funciona como una entidad lógica, controlando el estado y la consistencia de sus nodos, así como el estado de todo el clúster. Esto permite mantener la integridad de los datos de manera más eficiente que con la replicación asíncrona, sin perder escrituras seguras en múltiples nodos al mismo tiempo.

Sin embargo, pueden producirse escenarios en los que el servicio de base de datos puede detenerse sin que ningún nodo pueda atender solicitudes. Estos escenarios se describen en las secciones siguientes.

El nodo 1 se detiene correctamente

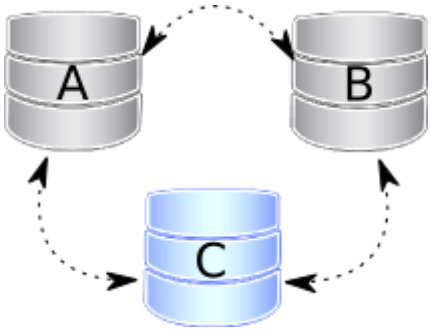


En un clúster de tres nodos (nodos 1, 2 y 3), el nodo 1 se detiene correctamente, con fines de mantenimiento, cambio de configuración, etc.

En este caso, los otros nodos reciben un mensaje de "adiós" del nodo detenido y el tamaño del clúster se reduce; algunas propiedades, como el cálculo de quórum o el incremento automático, se cambian automáticamente. Tan pronto como el nodo 1 se inicia de nuevo, se une al clúster en función de su variable wsrep_cluster_address en my.cnf.

Si la caché del conjunto de escritura (gcache.size) en los nodos 2 y/o 3 todavía tiene todas las transacciones ejecutadas mientras el nodo 1 estaba inactivo, la unión es posible a través de IST. Si IST es imposible debido a la falta de transacciones en el caché del donante, la decisión de reserva es tomada por el donante y SST se inicia automáticamente.

Dos nodos se detienen correctamente



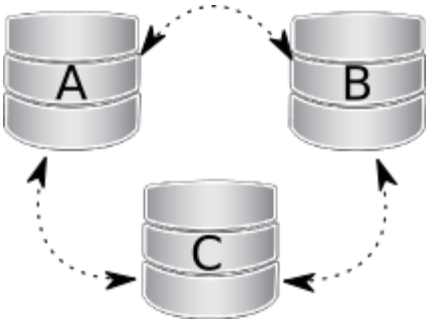
Como en el primer punto, el tamaño del clúster se reduce a 1, incluso el único nodo restante 3 forma el componente principal y puede atender las solicitudes del cliente. Para que los nodos vuelvan al clúster, solo tiene que iniciarlos.

Sin embargo, cuando un nuevo nodo se une al clúster, el nodo 3 cambiará al estado "Donante/Desincronizado", ya que tiene que proporcionar la transferencia de estado al menos al primer nodo de unión. Todavía es posible leer / escribir en él durante ese proceso, pero puede ser mucho más lento, lo que depende de la gran cantidad de datos que se deben enviar durante la transferencia de estado. Además, algunos equilibradores de carga pueden considerar que el nodo donante no está operativo y eliminarlo del grupo. Por lo tanto, es mejor evitar la situación cuando solo un nodo está activo.

Si reinicia el nodo 1 y, a continuación, el nodo 2, asegúrese de que el nodo 2 no utilice el nodo 1 como donante de transferencia de estado: es posible que el nodo 1 no tenga todos los conjuntos de escrituras necesarios en su gcache. Especifique el nodo 3 como donante en el archivo de configuración e inicie el servicio mysql:

$ systemctl start mysql

Los tres nodos se detienen correctamente



El clúster está completamente detenido y el problema es cómo inicializarlo de nuevo. Es importante que un nodo escriba su última posición ejecutada en el archivo grastate.dat.

Al comparar el número seqno en este archivo, puede ver cuál es el nodo más avanzado (probablemente el último detenido). El clúster debe arrancar utilizando este nodo, de lo contrario, los nodos que tenían una posición más avanzada tendrán que realizar el SST completo para unirse al clúster inicializado desde el menos avanzado. Como resultado, algunas transacciones se perderán).

Para arrancar el primer nodo, invoque el script de inicio de esta manera:

Para MySQL:

```
$ mysqld_bootstrap -- wsrep-new-cluster
```

Para PXC:

```
$ systemctl start mysql@bootstrap.service
```

Para MariaDB:

```
galera_new_cluster $
```

Nota

Aunque arranque desde el nodo más avanzado, los otros nodos tienen un número de secuencia más bajo. Todavía tendrán que unirse a través del SST completo, ya que la caché de Galera no se conserva al reiniciar. Por esta razón, se recomienda detener las escrituras en el clúster antes de su cierre completo, para que todos los nodos puedan detenerse en la misma posición. Véase también pc.recovery.

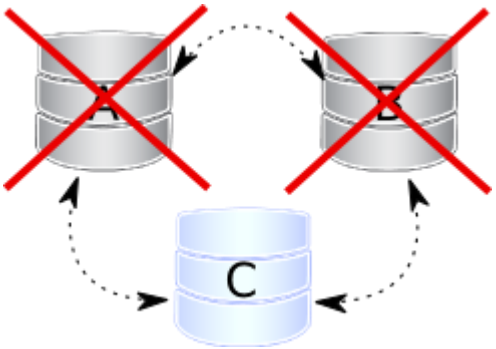
Un nodo desaparece del clúster

Este es el caso cuando un nodo deja de estar disponible debido, por ejemplo, a un corte de energía, falla de hardware, pánico del kernel, bloqueo de mysqld o kill -9 en mysqld pid.

Los dos nodos restantes notan que la conexión al nodo 1 está inactiva y comienzan a intentar volver a conectarse a él. Después de varios tiempos de espera, el nodo 1 se quita del clúster. El quórum se guarda (dos de cada tres nodos están activos), por lo que no se produce ninguna interrupción del servicio. Después de reiniciarse, el nodo 1 se une automáticamente, como se describe en El nodo 1 se detiene correctamente.

Dos nodos desaparecen del clúster

Dos nodos no están disponibles y el nodo restante (nodo 3) no puede formar el quórum solo. El clúster tiene que cambiar a un modo no primario, donde MySQL se niega a servir cualquier consulta SQL. En este estado, el proceso "mysqld" en el nodo 3 todavía se está ejecutando y se puede conectar, pero cualquier instrucción relacionada con los datos falla con un error.



MySQL> **Select** * **from** test.sbtest1;

ERROR 1047 (08S01): WSREP aún **no** ha preparado el nodo **para el uso** de la aplicación

Las lecturas son posibles hasta que el nodo 3 decida que no puede acceder al nodo 1 y al nodo 2. Las nuevas escrituras están prohibidas.

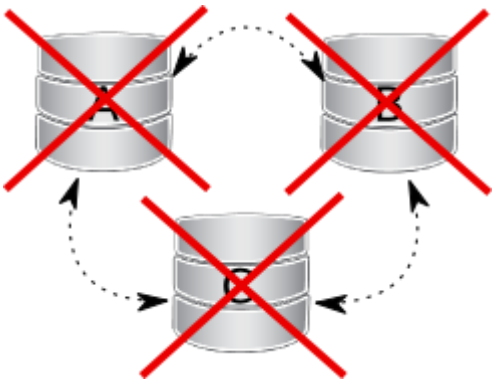
Tan pronto como los otros nodos estén disponibles, el clúster se vuelve a formar automáticamente. Si el nodo 2 y el nodo 3 simplemente se separaron de la red del nodo 1, pero aún pueden comunicarse entre sí, seguirán funcionando ya que aún forman el quórum.

Si el nodo 1 y el nodo 2 se bloquearon, debe habilitar el componente principal en el nodo 3 manualmente, antes de poder abrir el nodo 1 y el nodo 2. El comando para hacer esto es:

```
mysql> SET GLOBAL wsrep_provider_options='pc.bootstrap=true';
```

NOTA: Este enfoque solo funciona, si los otros nodos están inactivos antes de hacerlo de lo contrario, terminará con dos clústeres con datos diferentes.

Todos los nodos se caen sin un procedimiento de apagado adecuado



Este escenario es posible en el caso de un fallo de alimentación del centro de datos o cuando se produce un error de MySQL o Galera. Además, puede suceder como resultado de que la consistencia de los datos se vea comprometida cuando el clúster detecta que cada nodo tiene datos diferentes. El archivo grastate.dat no se actualiza y no contiene un número de secuencia válido (seqno). Puede verse así:

$ cat /var/lib/mysql/grastate.dat

Estado guardado de GALERA

Versión: 2.1

UUID: 220dcdcb-1629-11e4-add3-aec059ad3734

Seqno: -1

safe_to_bootstrap: 0

En este caso, no puede estar seguro de que todos los nodos sean coherentes entre sí. No podemos usar safe_to_bootstrap variable para determinar el nodo que tiene la última transacción confirmada, ya que se establece en 0 para cada nodo. Un intento de arranque desde dicho nodo fallará a menos que inicie mysqld con el parámetro --wsrep-recover:

$ mysqld --wsrep-recover

Busque en la salida la línea que informa de la posición recuperada después del UUID del nodo (1122 en este caso):

...

... [Nota] WSREP: Posición recuperada: 220dcdcb-1629-11e4-add3-aec059ad3734:1122

...

El nodo donde la posición recuperada está marcada por el mayor número es el mejor candidato de arranque. En su archivo grastate.dat, establezca la variable safe_to_bootstrap en 1. A continuación, arranque desde este nodo.

Nota

Después de un apagado, puede boostrap desde el nodo que está marcado como seguro en el archivo grastate.dat.

...

safe_to_bootstrap: 1

...

La opción pc.recovery (habilitada de forma predeterminada) guarda el estado del clúster en un archivo denominado gvwstate.dat en cada nodo miembro. Como sugiere el nombre de esta opción (pc – componente primario), solo guarda un clúster en el estado PRIMARIO. Un contenido de ejemplo de un archivo gvwstate.dat puede verse así:

cat /var/lib/mysql/gvwstate.dat

my_uuid: 76DE8AD9-2AAC-11E4-8089-D27FD06893B9

#vwbeg

view_id: 3 6C821ECC-2AAC-11E4-85A5-56FE513C651F 3

Arranque: 0

Miembro: 6C821ECC-2AAC-11E4-85A5-56FE513C651F 0

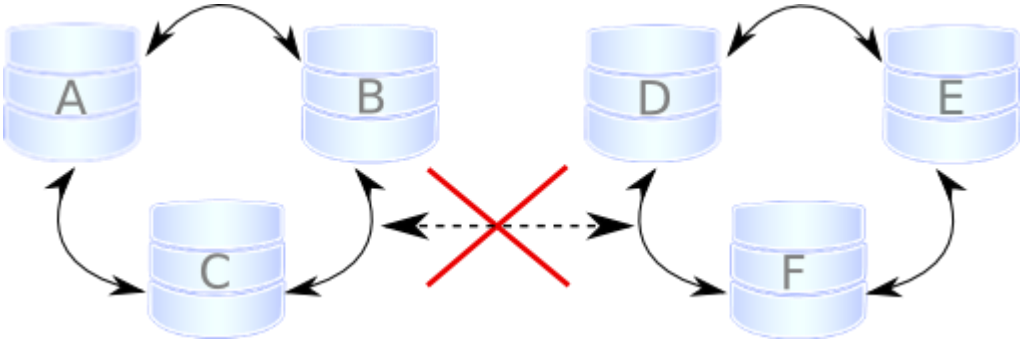
Miembro: 6D80EC1B-2AAC-11E4-8D1E-B2B2F6CAF018 0

Miembro: 76DE8AD9-2AAC-11E4-8089-D27FD06893B9 0

#vwend

Podemos ver un clúster de tres nodos con todos los miembros activos. Gracias a esta característica, los nodos intentarán restaurar el componente principal una vez que todos los miembros comiencen a verse entre sí. ¡Esto hace que el clúster se recupere automáticamente de ser apagado sin ninguna intervención manual!

El cluster pierde su estado primario debido a la “división del cerebro”



Supongamos que tenemos un clúster que consta de un número par de nodos: seis, por ejemplo. Tres de ellos están en una ubicación, mientras que los otros tres están en otra ubicación y pierden la conectividad de red. Se recomienda evitar dicha topología: si no puede tener un número impar de nodos reales, puede utilizar un nodo arbitrador adicional (garbd) o establecer un pc.weight más alto para algunos nodos. Pero cuando el “cerebro dividido” ocurre de alguna manera, ninguno de los grupos separados puede mantener el quórum: todos los nodos deben dejar de atender solicitudes y ambas partes del clúster intentarán continuamente volver a conectarse.

Si desea restaurar el servicio incluso antes de que se restaure el vínculo de red, puede volver a hacer que uno de los grupos sea primario con el mismo comando que se describe en Dos nodos desaparecen del clúster.

SET GLOBAL wsrep_provider_options='pc.bootstrap=true';

Después de esto, puede trabajar en la parte restaurada manualmente del clúster, y la otra mitad debería poder volver a unirse automáticamente usando IST, tan pronto como se restaure el enlace de red.

Advertencia

Si establece la opción de arranque en ambas partes separadas, terminará con dos instancias de clúster vivo, con datos que probablemente diverjan entre sí. La restauración de un enlace de red en este caso no hará que se vuelvan a unir hasta que se reinicien los nodos y los miembros especificados en el archivo de configuración se vuelvan a conectar. Luego, como el modelo de replicación de Galera realmente se preocupa por la consistencia de los datos: una vez que se detecta la inconsistencia, los nodos que no pueden ejecutar la instrucción de cambio de fila debido a una diferencia de datos, se realizará un apagado de emergencia y la única forma de devolver los nodos al clúster es a través del SST completo.

Este artículo se basa en la entrada de blog Replicación de Galera - cómo recuperar un clúster PXC por Przemysław Malkowski: Replicación de Galera: cómo recuperar un clúster de PXC

Sobre Virtual Cable

Virtual Cable es una compañía especializada en la **transformación digital** del **puesto de trabajo**. La compañía desarrolla, soporta y comercializa UDS Enterprise. Su equipo de expertos ha diseñado soluciones **VDI** a medida de **cada sector** para proporcionar una experiencia de usuario única y totalmente adaptada a las necesidades de cada perfil de usuario. Los profesionales de Virtual Cable tienen **más de 30 años de experiencia** en TI y desarrollo de software y más de 15 en tecnologías de virtualización. Cada día se despliegan **millones de escritorios virtuales Windows y Linux con UDS Enterprise en todo el mundo**.