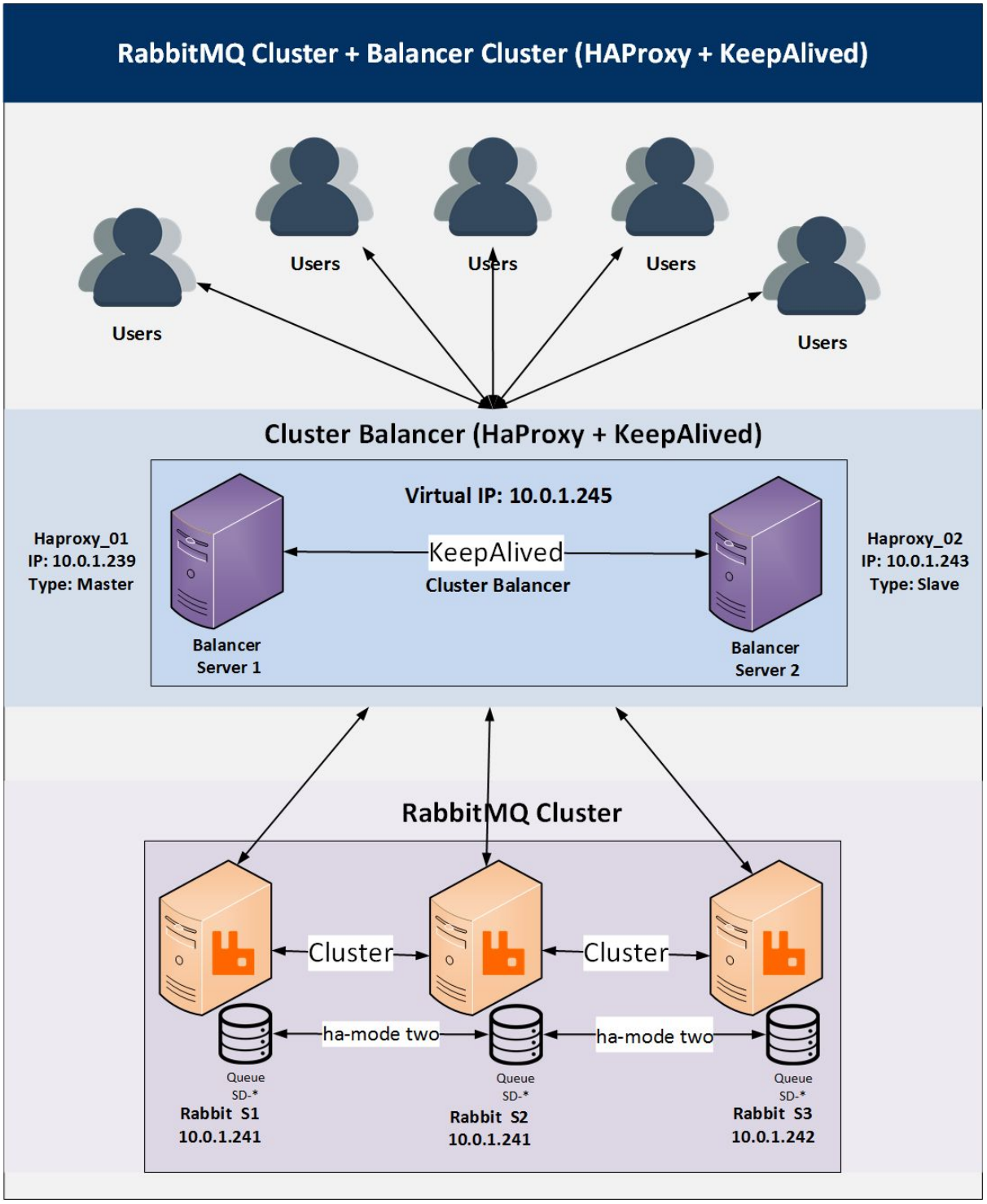
Balanceador en Cluster (HAProxy + KeepAlived) + Cluster de RabbitMQ



Paso 1: Instalación y Configuración de RabbitMQ

#Si estamos detrás de proxy, utilizar export en la terminal para salir a internet
export http_proxy="http://domain\user@userPassword@ip:port”
#nano /etc/apt/sources.list

- Colocar los repositorios y actualizar a última versión

Oficiales
deb http://ftp.us.debian.org/debian/ stretch main contrib non-free
deb-src http://ftp.us.debian.org/debian/ stretch main contrib non-free

Actualizaciones de seguridad
deb http://security.debian.org/debian-security stretch/updates main contrib non-free
deb-src http://security.debian.org/debian-security stretch/updates main contrib non-free

---

- Actualizar la versión del SO

#apt-get update
#apt-get dist-upgrade

- Verificar ahí que la versión esté actualizada

#lsb_release -a

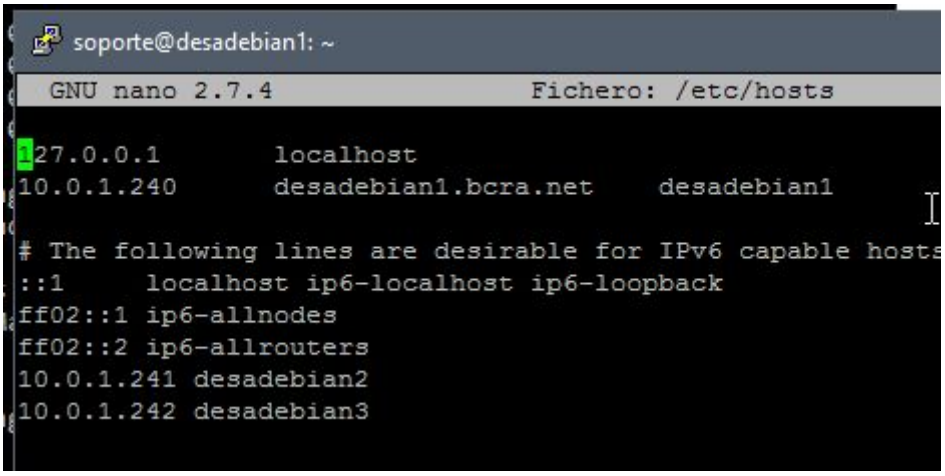
```
root@desadebian1:~# lsb_release -a
No LSB modules are available.
Distributor ID: Debian
Description:    Debian GNU/Linux 9.9 (stretch)
Release:        9.9
Codename:       stretch
```

- Luego verificar que el nombre del host esté cargado correctamente:

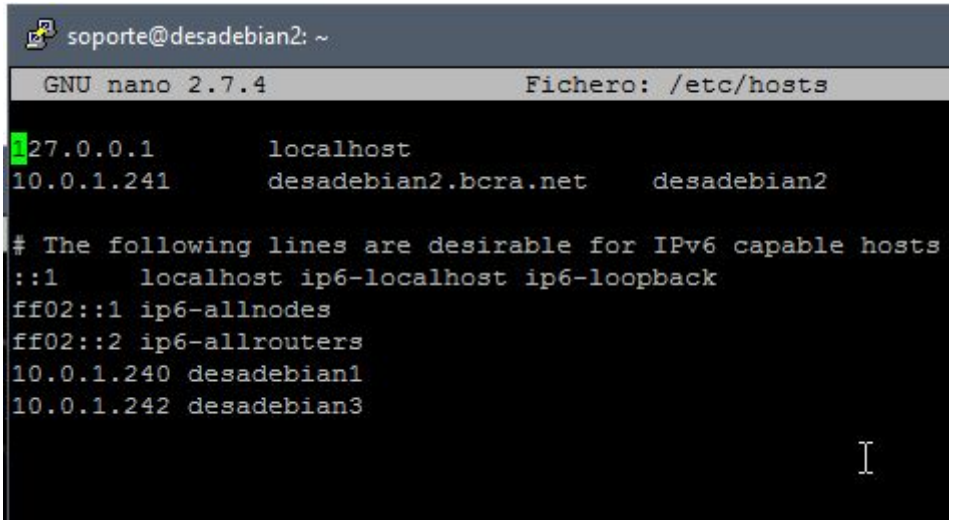
#nano /etc/hostname

- Ahora tenemos que agregar los nombres e IPs de los otros servidores para que en la configuración del cluster se puedan ver (tres servers en este caso)

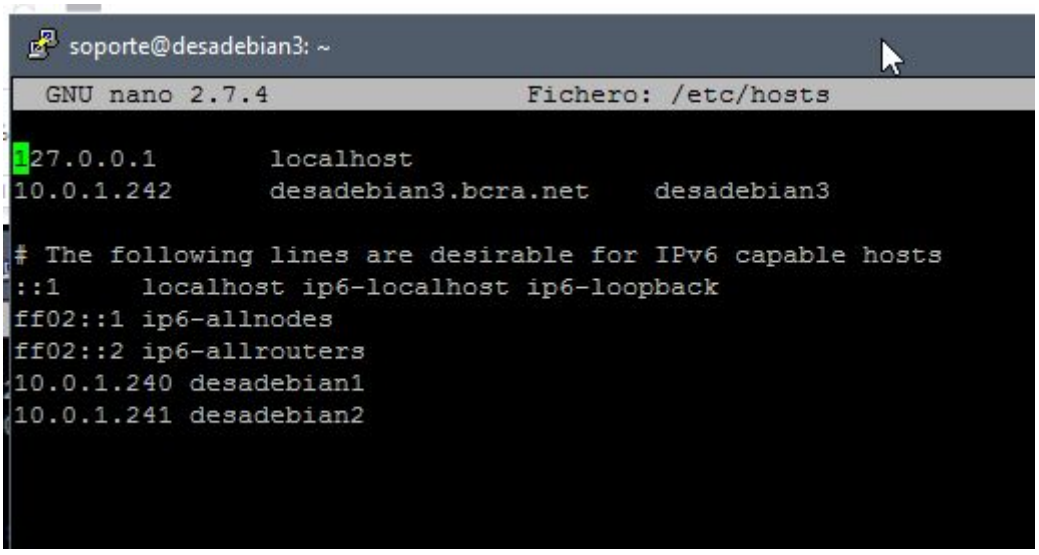
#nano /etc/hosts
desadebian1 (Rabbit Server 1):



desadebian2 (Rabbit Server 2):



desadebian3 (Rabbit Server 3):



- Reiniciar equipos para que se apliquen los cambios

#reboot

- Instalar rabbitmq-server en ambos equipos

(Si proxy, hacer export de nuevo)
#apt-get install rabbitmq-server

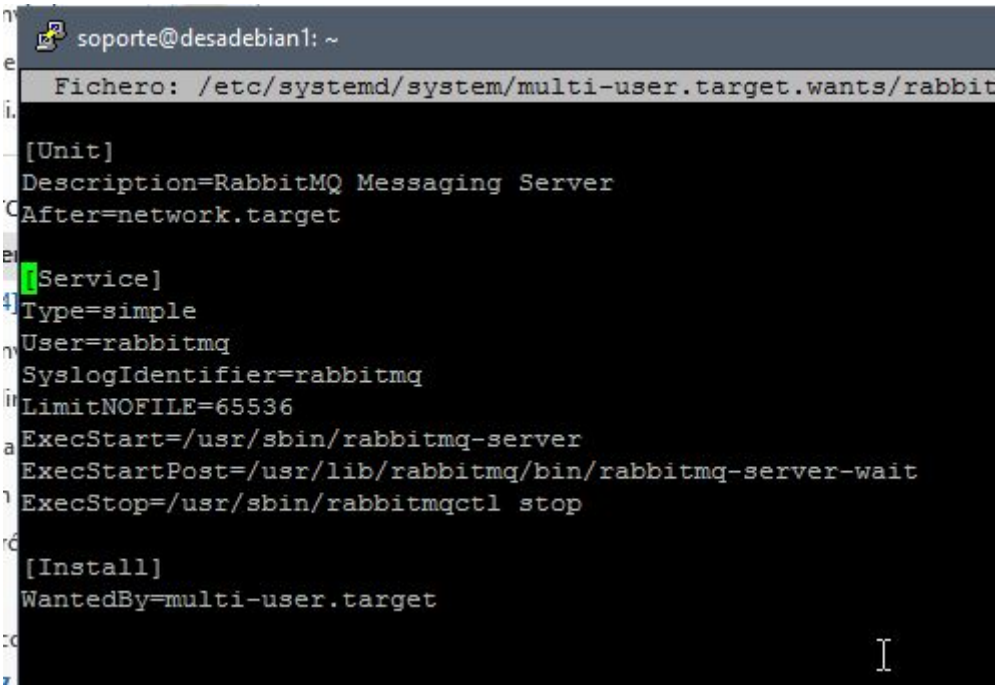
- Habilitar administración web, usuarios por defecto con acceso al virtual host y federación

#rabbitmq-plugins enable rabbitmq_management
#rabbitmq-plugins enable rabbitmq_federation
#rabbitmq-plugins enable rabbitmq_federation_management
#rabbitmqctl add_user admin admin
#rabbitmqctl set_user_tags admin administrator
#rabbitmqctl set_permissions -p / admin ".*" ".*" ".*"

- Ahora podemos irnos a un navegador web y poner la dirección IP de uno de los nodos junto al puerto 15672 y vemos cómo podemos acceder a la administración de RabbitMQ, por ejemplo en nodo 1

- Por defecto nuestro sistema se configura en modo desarrollo con un **File Descriptor = 1024**. Para entornos de producción se recomienda al menos aumentar ese valor a **65536**.

#nano /etc/systemd/system/multi-user.target.wants/rabbitmq-server.service



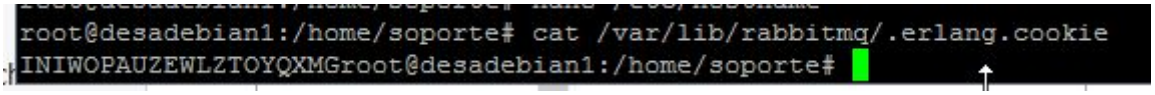
- Como las peticiones las vamos a realizar a través de un proxy, si en la web de RabbitMQ queremos ver la dirección IP del cliente que realiza la conexión, en vez de la dirección IP del proxy necesitamos añadir un setting **(proxy_protocol = true)** en el fichero de configuración **rabbitmq.conf**.

#nano /etc/rabbitmq/rabbitmq.conf
-> escribir proxy_protocol = true, luego guardar
#systemctl daemon-reload
#service rabbitmq-server restart

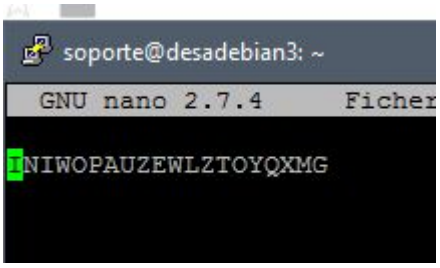
Paso 2: Configuración del Clúster RabbitMQ

- Para formar el clúster y que todos los nodos funcionen en conjunto es necesario que **todos tengan la misma cookie de Erlang**. Para ello entramos en el nodo principal que en este caso será **rabbit_01** y consultamos su cookie. Ésta la deberemos copiar a los otros dos nodos para que sea la misma.

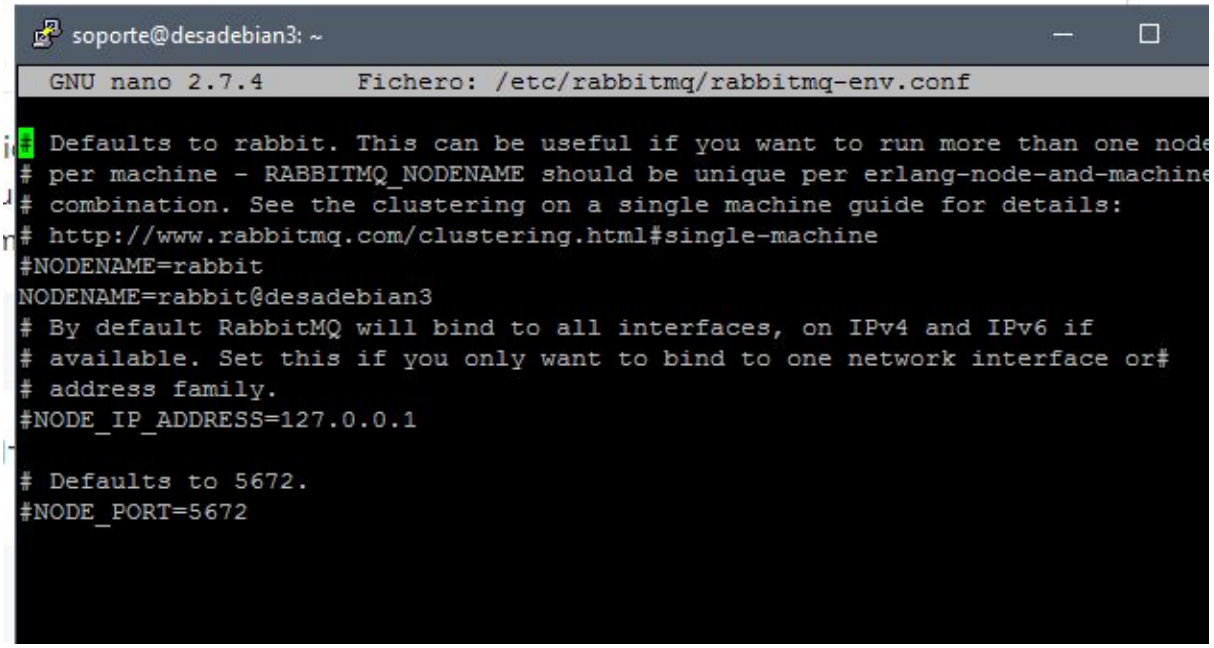
Verificar Earlang en desadebian1
#cat /var/lib/rabbitmq/.erlang.cookie



y cargarla en los otros nodos
#nano /var/lib/rabbitmq/.erlang.cookie



- Ahora deberemos configurar el archivo rabbitmq-env.conf de cada nodo donde pondremos el nombre del nodo. Este estará formado por rabbit@[hostname], donde rabbit es el nombre del cluster



Importante: Ver (como dice la documentación) que el sistema básico de Rabbit trabaja con el puerto tcp 5672

#nano /etc/rabbitmq/rabbitmq-env.conf
#service rabbitmq-server restart
(en cada server)

- Ahora revisar el estado del cluster (debería estar solamente el nodo1 en el cluster)

#rabbitmqctl cluster_status

```
Cluster status of node rabbit@rabbit_01 ...
[{nodes,[{disc,[rabbit@rabbit_01]}]},{running_nodes,[rabbit@rabbit_01]}]
...done.
```

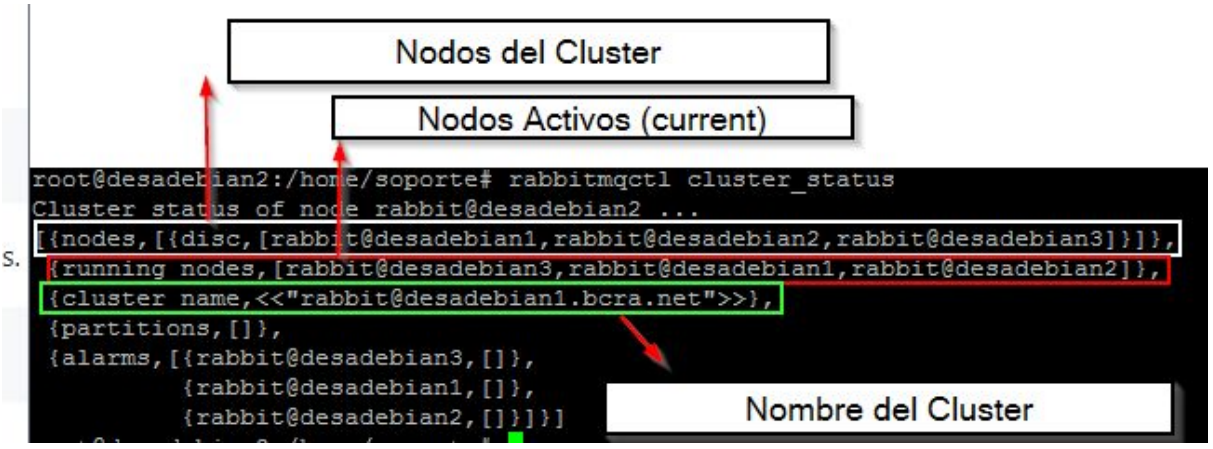
- Ahora unimos los nodos al cluster. Empezamos por el nodo 2 (desadebian2) con el nodo 1 (desadebian1) **que será el maestro.**

(Desde desadebian2)
#rabbitmqctl stop_app
#rabbitmqctl join_cluster rabbit@desadebian1
#rabbitmqctl start_app

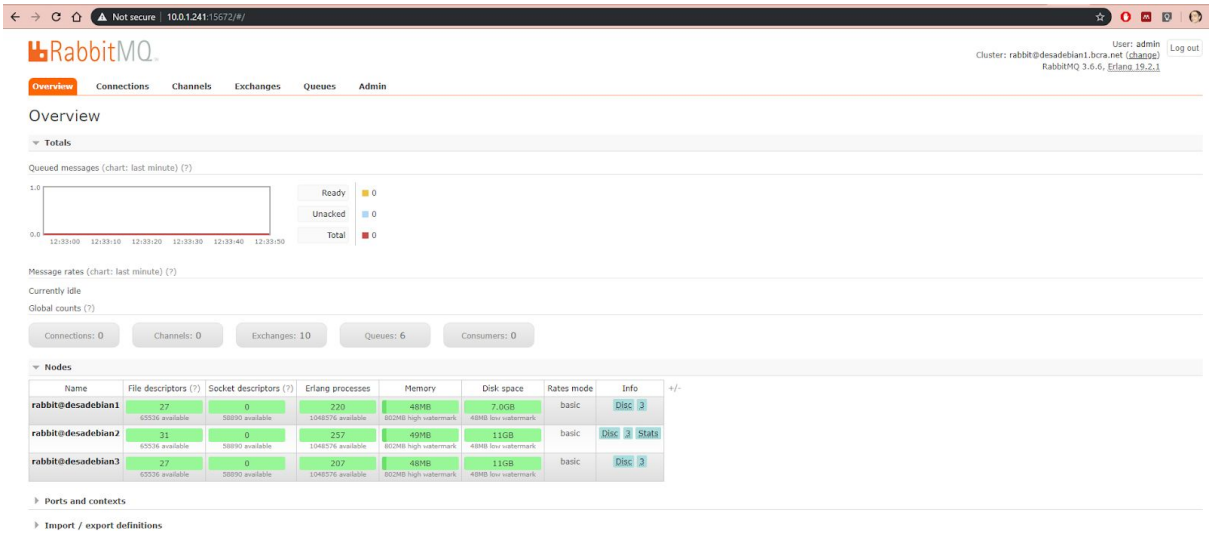
A continuación, unimos el nodo 3 (desadebian3) con el nodo 2 (desadebian2)

(Desde desadebian3)
#rabbitmqctl stop_app
#rabbitmqctl join_cluster rabbit@desadebian2
#rabbitmqctl start_app

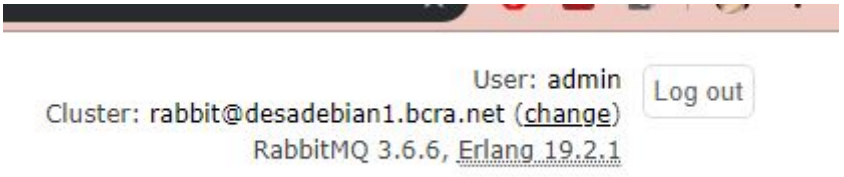
- Verificamos nuevamente el resultado del estado del cluster



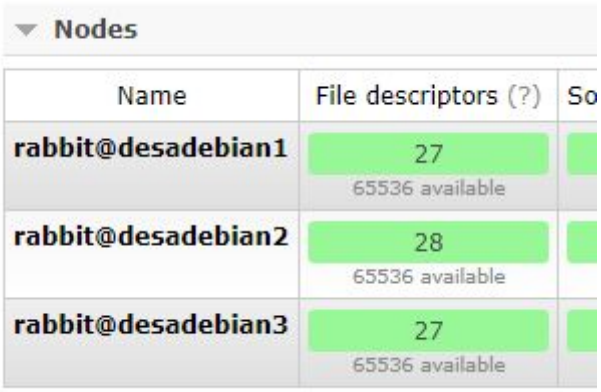
- Si vamos a la web de uno de los RabbitMQ podemos observar cómo nos aparece la información de los tres nodos (Cualquiera, por ejemplo el nodo 2)



Analizar en información por partes:



ClusterName: rabbit@desadebian1.bcra.net

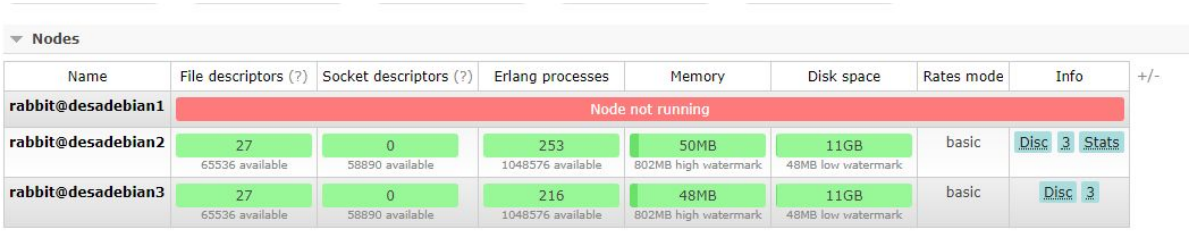


Nodes: Nuestros 3 nodos activos (verde)

- Verificar que si hago stop de uno de los nodos, el estado del cluster se refresca. Por ejemplo el nodo1 (desadebian1)

#rabbitmqctl stop_app

```
root@desadebian1:/home/soporte# cat /var/lib/rabbitmq/.erlang.cookie
INIWOPAUZEWLZTOYQXMGroot@desadebian1:/home/soporte# rabbitmqctl stop_app
Stopping node rabbit@desadebian1 ...
```

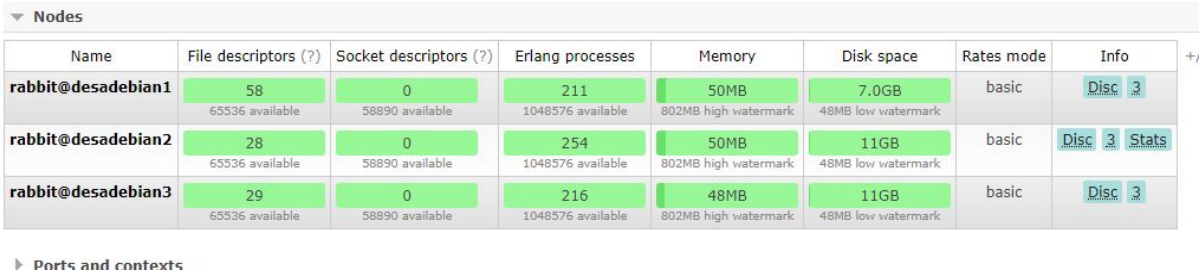


De esa manera automáticamente la herramienta del cluster nos muestra que el nodo principal no está activo. A su vez, con esto podemos verificar que el cluster sigue funcionando.

- Iniciamos nuevamente el servicio del nodo1 (desadebian1) que paramos

#rabbitmqctl start_app

Nuevamente vemos como se actualiza la información



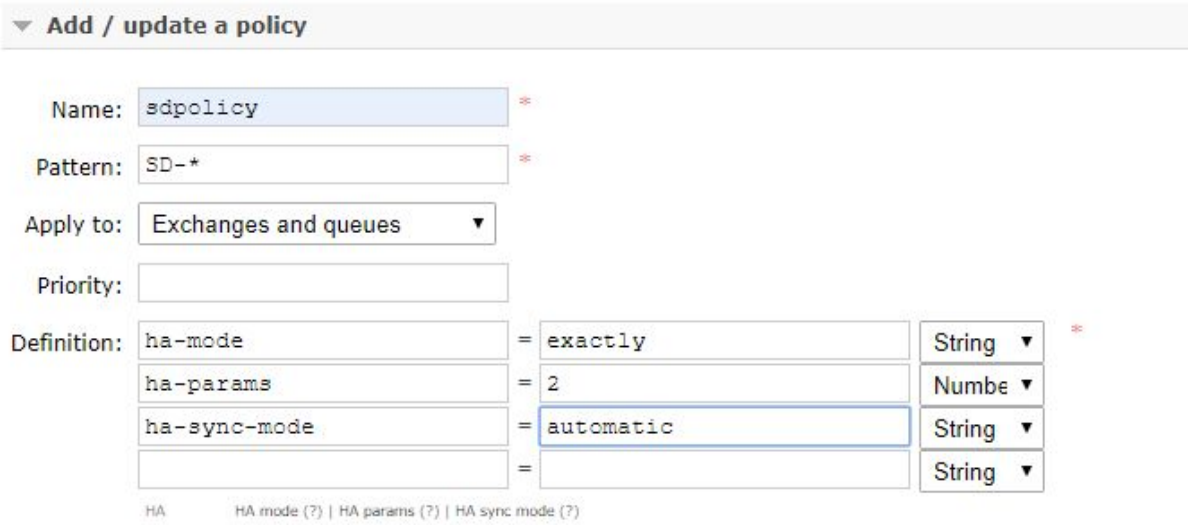
- Una vez que el cluster de rabbitmq está creado, falta configurar que las colas se repliquen a lo largo de los nodos según las políticas (policy) que se definen.

Pasos:

Admin > Policies > Add / update a policy.

- En el formulario, cargar los siguientes datos y confirmar

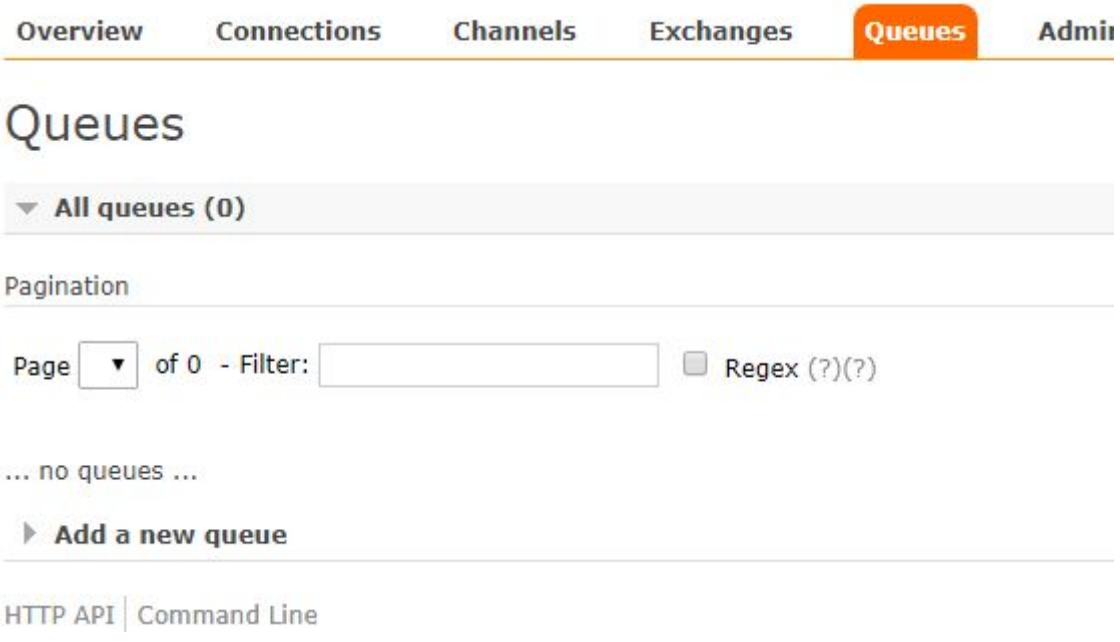
Name: sdpolicy
Pattern: SD-*
Apply to: Exchanges and Queues
Definition
ha-mode = exactly (string)
ha-params = 2 (number)
ha-sync-mode = automatic (string)



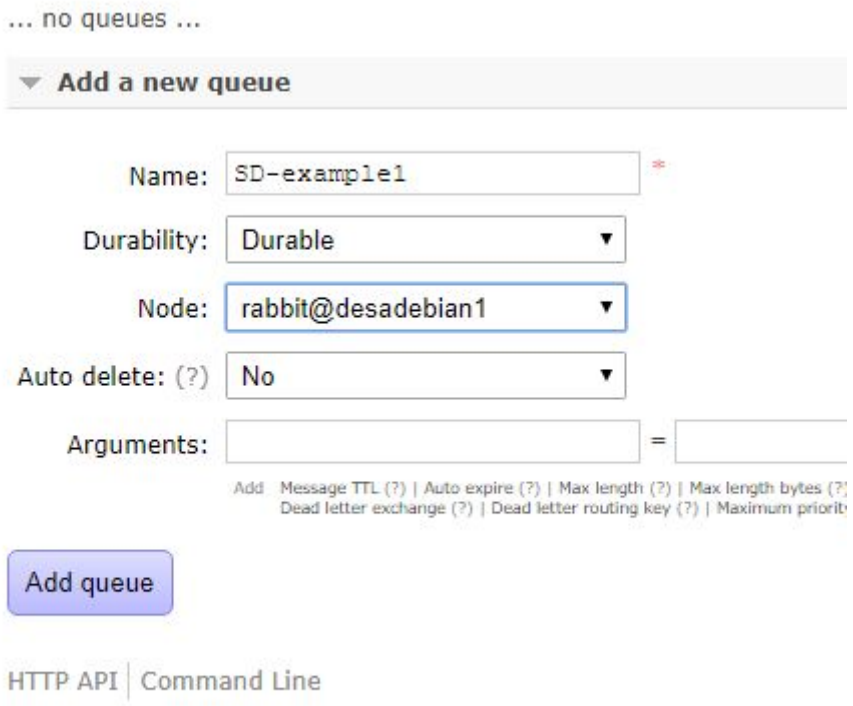
- Una vez finalizada y cargada la regla revisar que haya quedado correctamente definida en el mismo sitio web.

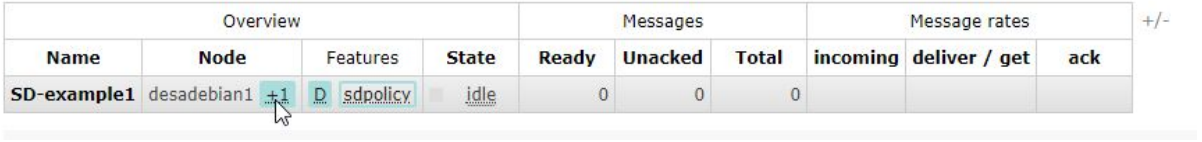
sdpolicy	SD-*	all	ha-mode: exactly ha-params: 2 ha-sync-mode: automatic	0

- Una vez terminados los pasos de la definición de la regla de las colas, vamos a verificar que no exista ninguna cola en la red



- Ahora vamos a verificar que cuando creamos una nueva cola con un nombre que empiece con SD- se cumple la condición y analicemos los destinos donde se encuentra replicado





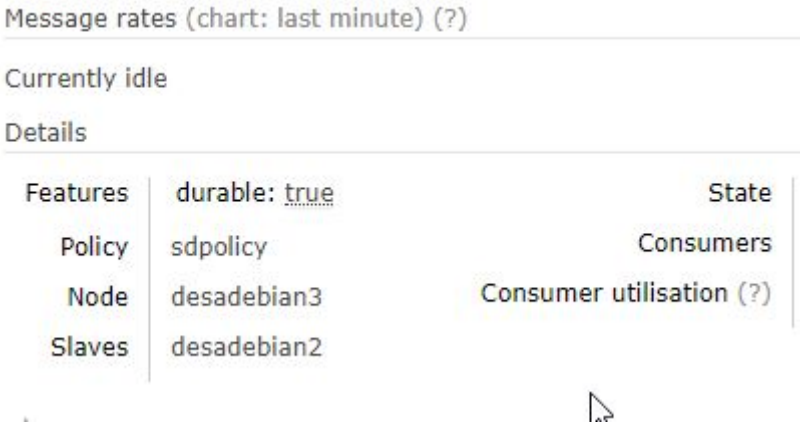
Overview				Messages			Message rates			+/-
Name	Node	Features	State	Ready	Unacked	Total	incoming	deliver / get	ack	
SD-example1	desadebian1 +1	D sdpolicy	idle	0	0	0				

Se puede visualizar como la cola SD-example, aplica la política sdpolicy, y por lo tanto más allá de estar disponible en desadebian1 (dónde se creó), existe un +1 que significa que esa cola se replicó en otro server (desadebian3) en este caso.

- Procedo a parar el nodo1 (desadebian1) y deberíamos poder ver los datos y/o escribir.

#rabbitmqctl stop_app

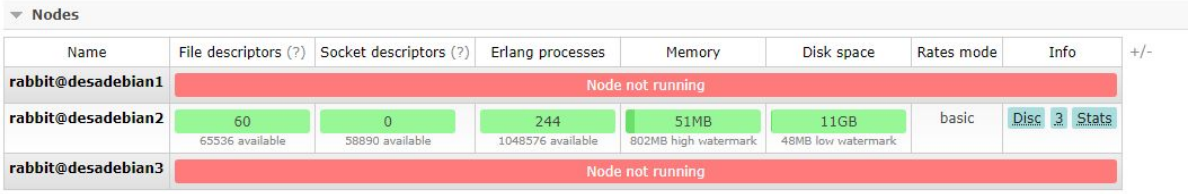
vamos al sitio y vemos que movió de nodo la cola (nodo principal) nodo1 a nodo3.



- Ahora insertar un mensaje y validar que es lo que pasa cuando se mueve (desde herramienta web). Luego se hará desde Java.

Overview				Messages			Message rates			+/
Name	**Node**	Features	**State**	**Ready**	**Unacked**	**Total**	**incoming**	**deliver / get**	**ack**	
SD-example1	desadebian3 +1	D sdpolicy	idle	1	0	1	0.00/s	0.00/s	0.00/s	

- Paro los dos servidores donde estaba la cola (nodo3 maestro, nodo1 replica), dejando solo el servidor dos (desadebian2).

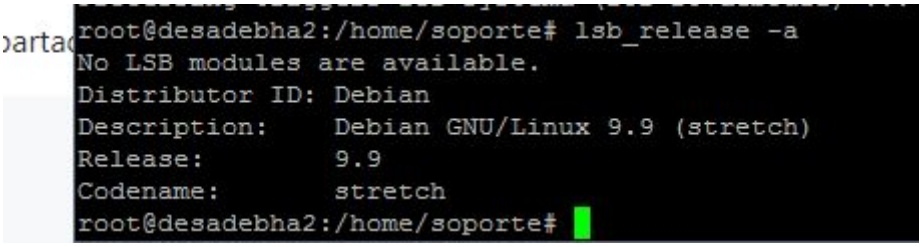


- Verificamos de igual manera que la cola se movió al server correspondiente (nodo2) y aún tenemos servicio

Overview				Messages			Message rates			+/-
Name	**Node**	Features	**State**	**Ready**	**Unacked**	**Total**	**incoming**	**deliver / get**	**ack**	
SD-example1	desadebian2	D sdpolicy	idle	1	0	1				

Paso 3: Configuración del Balanceador en Clúster

- En los equipos correspondientes al servicio de balanceador, realizar los primeros pasos de actualización y upgrade de los equipos como se explicó anteriormente, hasta llegar a validar repositorios y última distro de debian.



3.1 Instalación HAProxy

- Instalamos haproxy

#apt-get install haproxy

- Luego de instalado se debe configurar el haproxy.cfg. Primero se debe dar permiso a la carpeta para modificar el archivo de configuración. Luego, hacer una copia del archivo actual, y finalmente editar el template básico.

#chmod 777 /etc/haproxy
#cp /etc/haproxy/haproxy.cfg /etc/haproxy/haproxy_old.cfg
#nano /etc/hayproxy/haproxy.cfg

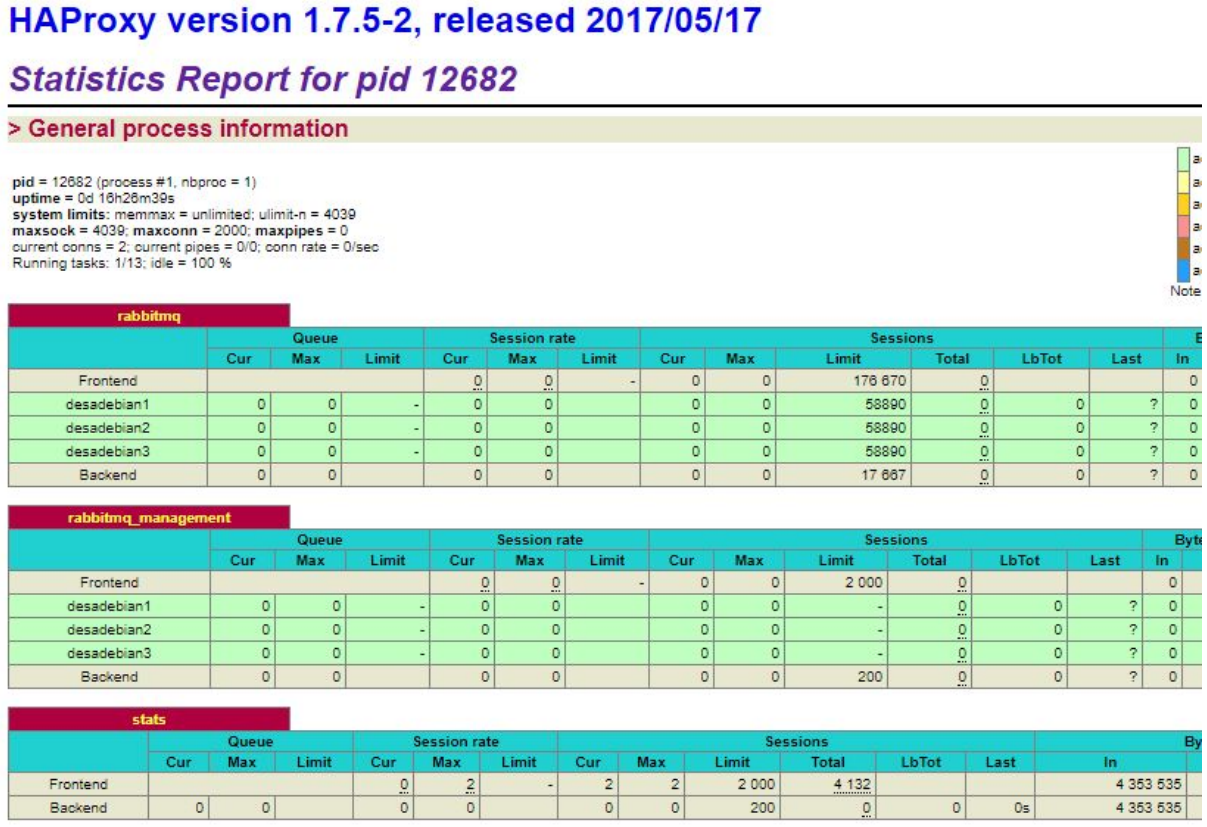
Agregar al archivo de configuración que está por defecto las siguientes líneas que tienen que ver con la configuración del balanceo y stats de rabbitmq

```
listen rabbitmq
        #todo lo que venga a 5670
bind 0.0.0.0:5672

mode tcp
        #Lo balanceo con mismo peso (RoundRobin)
balance roundrobin
        #Evitar desconexiones de los CLI se configura timeout client/server 3h
timeout client 3h
timeout server 3h
        #Configuracion clitcpka -> enviarse paquetes de Heartbeat (Cliente) y no se pierda conexion
option clitcpka

server rabbitmaster 10.0.1.240:5672 check inter 5s rise 2 fall 3
server rabbitmaster 10.0.1.241:5672 check inter 5s rise 2 fall 3
server rabbitmaster 10.0.1.242:5672 check inter 5s rise 2 fall 3
```

```
listen rabbitmq_management
        #todo lo que venga a 15672
    bind 0.0.0.0:15672
    mode tcp
        #Lo balanceo con mismo peso (RoundRobin)
    balance roundrobin
    server desadebian1 10.0.1.240:15672 check fall 3 rise 2
    server desadebian2 10.0.1.241:15672 check fall 3 rise 2
    server desadebian3 10.0.1.242:15672 check fall 3 rise 2

listen stats
        #Interfaz de administracion
    bind 0.0.0.0:8181
    stats enable
    stats uri /
    stats realm Haproxy\ Statistics
    stats auth admin:admin
    stats refresh 5s
```

- Esto hay que hacerlo en ambos servidores (desadebbha1 y desadebha2)

La interfaz de administración se puede ver de la siguiente manera



HAProxy version 1.7.5-2, released 2017/05/17

Statistics Report for pid 12682

> General process information

pid = 12682 (process #1, nbproc = 1)
uptime = 0d 16h26m39s
system limits: memmax = unlimited; ulimit-n = 4039
maxsock = 4039; **maxconn** = 2000; **maxpipes** = 0
current conns = 2; current pipes = 0/0; conn rate = 0/sec
Running tasks: 1/13; idle = 100 %

Note

rabbitmq

	Queue			Session rate			Sessions						E
	Cur	Max	Limit	Cur	Max	Limit	Cur	Max	Limit	Total	LbTot	Last	In
Frontend				0	0	-	0	0	176 670	0			0
desadebian1	0	0	-	0	0		0	0	58890	0	0	?	0
desadebian2	0	0	-	0	0		0	0	58890	0	0	?	0
desadebian3	0	0	-	0	0		0	0	58890	0	0	?	0
Backend	0	0		0	0		0	0	17 667	0	0	?	0

rabbitmq_management

	Queue			Session rate			Sessions						Byt
	Cur	Max	Limit	Cur	Max	Limit	Cur	Max	Limit	Total	LbTot	Last	In
Frontend				0	0	-	0	0	2 000	0			0
desadebian1	0	0	-	0	0		0	0	-	0	0	?	0
desadebian2	0	0	-	0	0		0	0	-	0	0	?	0
desadebian3	0	0	-	0	0		0	0	-	0	0	?	0
Backend	0	0		0	0		0	0	200	0	0	?	0

stats

	Queue			Session rate			Sessions						By
	Cur	Max	Limit	Cur	Max	Limit	Cur	Max	Limit	Total	LbTot	Last	In
Frontend				0	2	-	2	2	2 000	4 132			4 353 535
Backend	0	0		0	0		0	0	200	0	0	0s	4 353 535

3.2 Instalación KeepAlived

- Ahora hay que instalar en ambos eqiupos keepalived

#apt-get install keepalived

- Luego de instalar hay que configurar el producto (con diferentes configuraciones en los equipos debido a uno se configurará como Maestro y otro como Esclavo.

- interface: donde indicamos la tarjeta de red de nuestro servidor.
- state: le decimos cual va a ser el MASTER y cual el BACKUP.
- priority: daremos más prioridad al servidor maestro de tal forma que en caso de que los dos servidores HAProxy estén iniciados tomará toda la carga de conexiones.
- virtual_router_id: identificador numérico que tiene que ser igual en los dos servidores.
- auth_pass: especifica la contraseña utilizada para autenticar los servidores en la sincronización de failover.
- virtual_ipaddress: será la dirección IP virtual que compartirán lo dos servidores y a la que tienen que realizar las peticiones los clientes.

Configuración de Maestro:
cp /etc/keepalived/keepalived.conf /etc/keepalived/keepalived_old.conf
nano /etc/keepalived/keepalived.conf

MAESTRO

```
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 ens192
virtual_router_id 51
priority 101
advert_int 1
# The virtual ip address shared between the two loadbalancers
```

virtual_ipaddress {
10.0.1.245
}
track_script {
check_haproxy
}
}

- Levantar servicio de keepalived

#service keepalived start
#service keepalived status

```
root@desadebha2:/home/soporte# service keepalived status
● keepalived.service - Keepalive Daemon (LVS and VRRP)
   Loaded: loaded (/lib/systemd/system/keepalived.service; enabled; vendor prese
   Active: active (running) since Wed 2019-05-29 07:13:16 CDT; 29min ago
  Process: 13737 ExecStart=/usr/sbin/keepalived $DAEMON_ARGS (code=exited, statu
 Main PID: 13738 (keepalived)
    Tasks: 3 (limit: 4915)
   CGroup: /system.slice/keepalived.service
           ├─13738 /usr/sbin/keepalived
           ├─13739 /usr/sbin/keepalived
```

- Revisión de que se haya asignado la ip virtual en el maestro

#ip addr list

```
root@desadebha1:/etc/keepalived# ip addr list
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1
    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: ens192: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq state UP group default qlen 1000
    link/ether 00:50:56:9a:93:d0 brd ff:ff:ff:ff:ff:ff
    inet 10.0.1.239/24 brd 10.0.1.255 scope global ens192
       valid_lft forever preferred_lft forever
    inet 10.0.1.245/24 scope global secondary ens192
       valid_lft forever preferred_lft forever
    inet6 fe80::250:56ff:fe9a:93d0/64 scope link
       valid_lft forever preferred_lft forever
```

IP Equipo

IP Virtual

ESCLAVO

---

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 BACKUP
interface ens192
virtual_router_id 51
priority 100
advert_int 1
# The virtual ip address shared between the two loadbalancers
virtual_ipaddress {
10.0.1.245
}
track_script {
check_haproxy
}
}
```

- Crear un restarter de los servicios de haproxy y keepalived para que tome los datos

```
#cd /home/soporte
#nano starter.sh
```

y en el contenido colocar

```
/etc/init.d/haproxy restart
/etc/init.d/keepalived restart
```

- Luego correr el SH generado con

```
#sh starter.sh
```

- Ahora realizar diferentes pruebas para bajar y subir los servicios de keepalived y haproxy.