Proyecto

parte 2

Asignatura: Administración de sistemas 2
Fecha: 19/5/2020
Autor: Diego Marco Beisty, 755232

RESUMEN

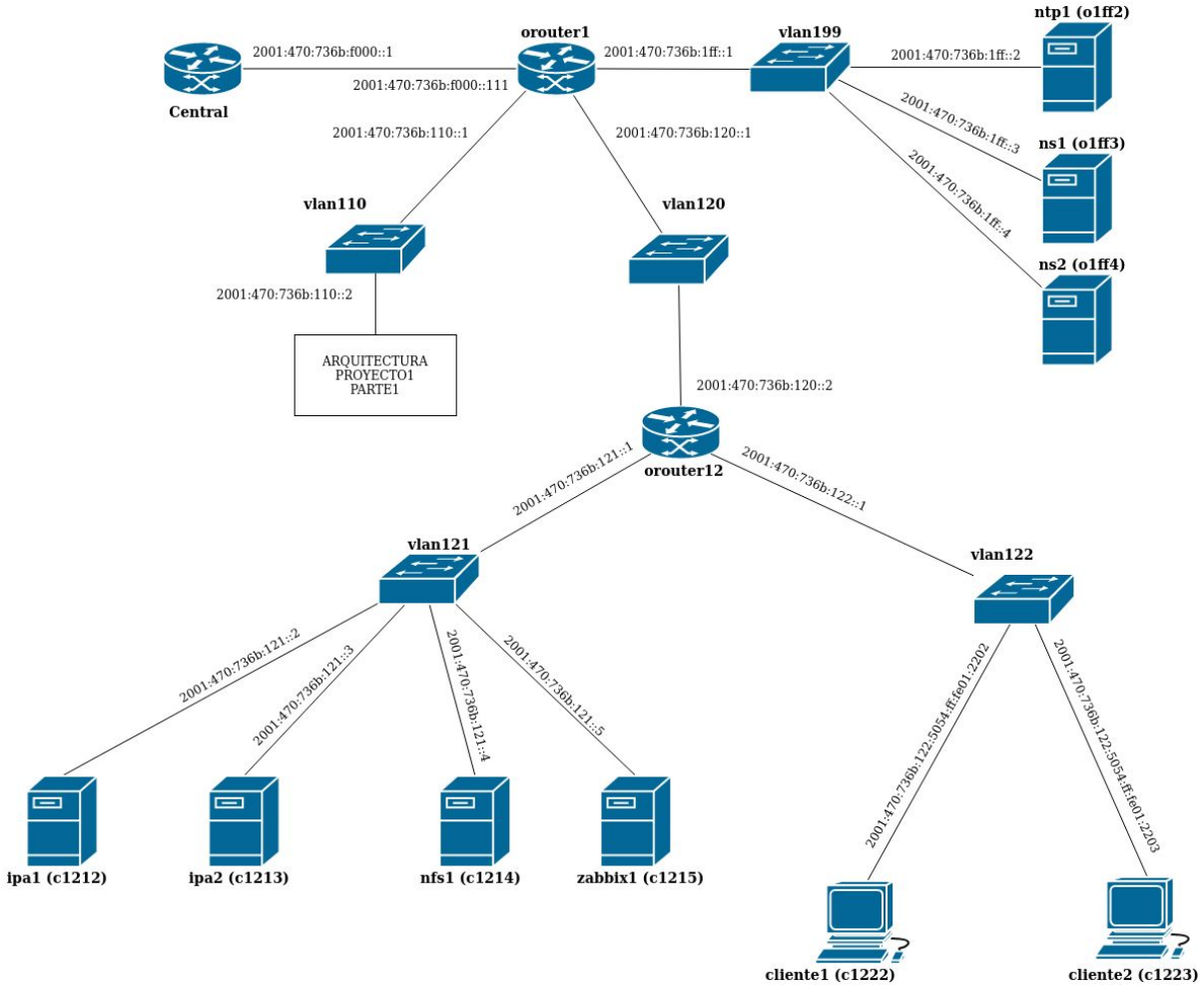
A partir de la arquitectura de la primera parte del proyecto, se crea una nueva subred vlan 120 formada por un router interno y otras dos subredes vlan 121 y 122. En estas subredes se configuran cuatro servidores y dos clientes. Estos siete nodos añadidos a la arquitectura han sido configurados mediante una estrategia de despliegue push con la herramienta Puppet.
Como en la primera parte del proyecto, dos servidores van a levantar un dominio freeIPA/DNS que cuelga del dominio DNS establecido en prácticas anteriores.
Se va a aprovechar la integración de servicios NTP, LDAP, Kerberos y DNS que supone freeIPA para sincronizar los relojes de las máquinas, centralizar cuentas de usuarios, cifrar las comunicaciones del sistema y ofrecer un servicio de resolución de nombres a los integrantes del dominio.
Además un tercer servidor se encargará de centralizar y compartir los directorios de los usuarios en las máquinas clientes mediante NFSv4.
Por último, el cuarto servidor se encargará de monitorizar el estado distribuido del sistema y avisar de posibles problemas, mediante el servicio zabbix.

INTRODUCCIÓN Y OBJETIVOS

En esta segunda parte del proyecto se plantea un despliegue automático de la configuración tanto a nivel de red, servicios básicos, servicio nfs y servicio zabbix.
Para ello se va a implementar un módulo puppet de red, uno de nfs y otro de zabbix que se instalarán en todos los nodos.
Después se elaborarán manifiestos puppet que declaren clases pertenecientes a los respectivos módulos con parámetros específicos para cada nodo.

El principal objetivo es aprender a usar módulos puppet.
Otro objetivo es utilizar la herramienta zabbix de forma avanzada con triggers, avisos, etc.

ARQUITECTURA DE ELEMENTOS RELEVANTES



Sin tener en los 7 nodos configurados en la parte 1 de este proyecto, en el diagrama quedan representados de arriba a abajo los siguientes elementos:

- El router Central que conecta las MVs a internet.
- orouter1, que conecta la subred externa 2001:470:736b:f000::/52 con las subredes vlan 199 2001;470:736b:1ff::/64, vlan 110 2001:470:736b:110::/60 y vlan 120 2001:470:736b:120::/60
- o1ff2(nombre DNS ntp1), es el servidor NTP y unbound.
- o1ff3(nombre DNS ns1), es el servidor primario DNS del dominio 1.ff.es.eu.org
- o1ff4(nombre DNS ns2), es el servidor secundario DNS del dominio 1.ff.es.eu.org
- orouter12 conecta la subred vlan 120, 2001:470:736b:120::/64 con las subredes vlan 121, 2001:470:736b:121::/64 y vlan 122, 2001:470:736b:122::/64.
- c1212(nombre DNS ipa1), es el servidor primario del dominio freeIPA/DNS 2.1.ff.es.eu.org.
- c1213(nombre DNS ipa2), es el servidor secundario del dominio freeIPA/DNS 2.1.ff.es.eu.org
- c1214(nombre DNS nfs1, es el servidor NFSv4.

- c1215(nombre DNS zabbix1), es el servidor de monitorización zabbix.
- c1222(nombre DNS cliente1), es el cliente 1 del sistema.
- c1223(nombre DNS cliente2), es el cliente 2 del sistema.

Cabe destacar que las 4 máquinas servidores y las 2 máquinas cliente tienen sistema operativo centOS y el router interno “orouter12” tiene S.O openBSD.

COMPREHENSIÓN DE ELEMENTOS RELEVANTES

Configuración de red

orouter1

Se ha configurado manualmente. Se ha añadido la nueva subred vlan 120 creando el fichero /etc/hostname.vlan120:

```
inet6 alias 2001:470:736b:110::1 64 vlan 110 vlandev vio0
   !route add -inet6 2001:470:736b:110::/60 2001:470:736b:110::2
   -autoconfprivacy
```

En la primera línea se observa la dirección ip de orouter1 en esta nueva subred junto con su máscara de red. Se indica también que la tarjeta vlan se crea sobre la tarjeta “física” vio0.
En la segunda línea se ha añadido una ruta estática para encaminar correctamente los paquetes que van dirigidos a las subredes vlan 111 y vlan 112.

orouter12

Se le ha automatizado una configuración parecida al nodo orouter11 de la parte 1 del proyecto.
Para ello se ha implementado la clase red::router_openbsd del módulo puppet red.
Esta clase se asegura que el estado del nodo sea el siguiente:
Tarjeta vio0 activa, sin ninguna dirección ipv6 asignada.
Tarjeta vlan 120 sobre vi0 configurada con la ip 2001:470:736b:120::2.
Router por defecto: 2001:470:736b:120::1.
Nombre del nodo: orouter12
Servidor DNS no autoritario: 2001:470:736b:1ff::2
Servidor NTP: 2001:470:736b:1ff::2
Tarjeta vlan 121 sobre vio0 configurada con la ip: 2001:470:736b:121::1
Tarjeta vlan 122 sobre vio0 configurada con la ip: 2001:470:736b:122::1
Servicio rad activo, sirviendo prefijos ipv6 a las subredes vlan 121 y 122.
Variable del kernel net.inet6.ip6.forwarding activada a 1 para que el nodo pueda enrutar paquetes.

Servidores ipa1, ipa2, nfs1, zabbix1
Se le ha automatizado una configuración parecida a los servidores de la parte 1 del proyecto.
Para ello se ha implementado la clase red::servidor_red del módulo puppet red.
Se ha declarado el siguiente estado para todos los servidores de la subred vlan 121:
La tarjeta eth0 tiene que estar encendida pero sin ninguna ipv6 asociada.
La tarjeta vlan 121 sobre eth0 tiene que existir y tener configurada una ip estática.
Las variables del kernel: net.ipv6.conf.eth0.use_tempaddr
net.ipv6.conf.eth0.autoconf
net.ipv6.conf.eth0.accept_ra
tienen que tener valor cero para evitar cualquier configuración dinámica de la ip.
Los servidores ipa1 e ipa2 tienen que tener como servidor DNS y NTP la ip 2001:470:736b:1ff::2 y los demás servidores las ips correspondientes a los servidores ipa: 2001:470:736b:121::2 ó 3.
La máquina tiene que tener un hostname asociado.

Clientes cliente1, cliente2
Se ha automatizado una configuración parecida a los clientes de la parte 1 del proyecto.
Para ello se ha implementado la clase red::cliente_red del módulo puppet red.
Se ha declarado el siguiente estado para todos los clientes de la subred vlan 122:
La tarjeta eth0 tiene que estar encendida pero sin ninguna ipv6 asociada.
La tarjeta vlan 122 sobre eth0 tiene que existir y tener configurada una ipv6 dinámica mediante el servicio rad del nodo orouter12.
Deben estar configurados como servidores NTP y DNS las máquinas ipa1 e ipa2: 2001:470:736b:121::2 y 2001:470:736b:121::3 respectivamente.
El nodo debe tener un hostname asociado.

Glue records DNS

Para poder añadir un nuevo dominio descendiente del ya configurado, es necesario añadir los registros denominados “glue records” en las zonas directas e inversas del dominio “1.ff.es.eu.org” del servidor DNS primario ns1 de la subred 1ff.
SIn los glue records cualquier máquina que pregunte por un nombre situado el subdominio “2.1.ff.es.eu.org” simplemente no obtendrá respuesta. Pero con los glue records, el servidor del dominio podrá responder con la IP del subdominio donde sí se encuentra el nombre DNS solicitado.
Fichero /var/nsd/zones/1.ff.es.eu.org.directo:

```
1       IN    NS    ipa1.1.1.ff.es.eu.org.
1       IN    NS    ipa2.1.1.ff.es.eu.org.
ipa1.2  IN    AAAA  2001:470:736b:121::2
ipa2.2  IN    AAAA  2001:470:736b:121::3
```

Fichero /var/zones/1.ff.es.eu.org.inverso:

```
2       IN    NS    ipa1.2.1.ff.es.eu.org.
```

```
2       IN    NS    ipa2.2.1.ff.es.eu.org.
```

Integración de servicios freeIPA

Dominio 2.1.ff.es.eu.org

Tras instalar el servidor freeIPA en ipa1 se han añadido las zonas directas e inversa del nuevo dominio. Además se han añadido todos los registros necesarios para resolver los nombres de dominio de los servidores y clientes. Este paso es necesario para apoyar la operativa de Kerberos.
Después se ha procedido a instalar el cliente freeIPA en las máquinas ipa2,nfs1,zabbix1,cliente1 y cliente 2.
Posteriormente en el servidor freeIPA se han añadido los principals de servicio nfs de las máquinas nfs1, cliente1 y cliente2. Los principals de host de estas máquinas no ha sido necesario añadirlos puesto que se añaden automáticamente al instalar los clientes freeIPA.
Finalmente se ha añadido en cada nodo los tickets kerberos de sesión para cada principal.

Cuentas de usuario freeIPA

Despueś de instalar el cliente freeIPA en el resto de máquinas, se ha añadido una cuenta para un usuario llamado “pepe”. Posteriormente se ha accedido a la cuenta desde la máquina cliente1 exitosamente.

Réplica freeIPA

Tras añadir la máquina ipa2 en el grupo ipaservers se ha procedido a instalar los paquetes software replica-install e ipa-server-dns para que ipa2 pueda actuar como réplica del dominio freeIPA.
Tras apagar el maestro freeIPA ,ipa1, se ha comprobado exitosamente que ipa2 adquiere el nuevo rol de maestro del dominio freeIPA.

Servicio NFS kerberizado

Exportación NFv4

Se ha automatizado la exportación nfsv4 en la máquina nfs1 mediante la clase nfs::servidor_nfs del paquete puppet nfs.
El estado en el que se pretende dejar esta máquina es el siguiente:
El paquete nfs-utils tiene que estar instalado.
El servicio nfs debe estar activo.
El fichero /etc/exports debe existir con el siguiente contenido:

```
/srv/nfs4/home  2001:470:736b:122::0/64(rw,sync,fsid=0,sec=krb5,no_subtree_check
```

Donde se indica en primer lugar la ruta del directorio que se exporta.
Después la subred a la que se da acceso al montaje de este pseudo sistema.

Por último las opciones de exportación entre las que se encuentran fsid=0, propia de nfsv4 y que indica que /srv/nfs4/home va a ser la raíz del pseudo sistema de ficheros que se monte en el cliente. Sec=krb5 indica que el montaje se va a realizar mediante Kerberos.

Montaje automático NFSv4

Se ha configurado con puppet el automontaje de los directorios de los clientes freeIPA en las máquinas cliente1 y cliente2. Para hecho se ha implementado la clase nfs::cliente_nfs del paquete nfs.
El estado en el que se pretende dejar a las máquinas clientes es el siguiente:
El paquete nfs-utils debe estar instalado.
El servicio autofs debe estar activo.
EL fichero /etc/auto.master debe tener asociado para el punto de montaje /home el mapa indirecto auto.home
El fichero /etc/auto.home debe tener el contenido:

```
*       -rw,fstype=nfs4,sec=krb5          nfs1.2.1.ff.es.eu.org:/&
```

Con el asterisco se captura cualquier directorio que se acceda bajo /home y si coincide con un directorio existente "&" que se exporte en el servidor remoto entonces se montará en el cliente.
La opción fstype=nfs4 obliga a hacer un montaje sobre nfsv4.
La opción sec=krb5 se incluye para que se monte utilizando kerberos.

Servicio zabbix

La configuración de este servicio se ha realizado mediante una automatización en bash.
Se han instalado en el servidor zabbix1 los servicios httpd y mariadb, necesarios para el frontend de zabbix y para guardar los datos recolectados de los agentes zabbix.
Después se ha poblado la base de datos con las tablas y registros que necesita inicialmente zabbix para poder operar.
Finalmente se han instalado los agentes zabbix en las máquinas zabbix1, ipa1, ipa2, nfs1, cliente1 y cliente2.
Se ha accedido remotamente al servicio mediante un navegador con la url: zabbix1.2.1.ff.es.eu.org/zabbix

PROBLEMAS ENCONTRADOS

Se han solucionado los problemas encontrados en la defensa de la primera parte del proyecto:
El primero consistía en que los usuarios freeIPA no podían cambiar la contraseña con el comando: passwd user.
Se ha descubierto que no era un problema como tal sino que el comando correcto para cambiar la contraseña era: passwd.
El problema se debe a que el primer comando passwd toma como parámetro un usuario y esto se interpreta como una acción que solo puede realizar un usuario root, que es cambiar la contraseña de un usuario cualquiera. Sin embargo con el segundo comando passwd, sin añadir ningún nombre de usuario, se entiende implícitamente que se quiere automodificar la contraseña y por lo tanto se tienen permisos para hacerlo.

El segundo problema consistía en que la réplica ipa2 de freeIPA no funcionaba.
Esto se debía a que faltaban de instalar los paquetes ipa-replica e ipa-server-dns.
Tras su instalación aparecieron dos problemas más.
La máquina se quedaba sin memoria al instalar estos paquetes, por lo que fue necesario aumentar un gigabyte más de memoria.
Además en el fichero /etc/pki/pki-tomcat/server.xml, en la línea 219, el parámetro address tenía asociado la ip 127.0.0.1 y debido a que la infraestructura configurada solo trabaja con ipv6 se cortaba la comunicación de la réplica con el servidor de certificados en la máquina ipa1. Se solucionó cambiando este parámetro con la direcctión ipv6 ::1.

Durante la realización de la parte dos del proyecto ha surgido un problema.
El paquete puppet zabbix de la web forge.puppet no era compatible con la versión de la máquina centOS, zabbix1 (3.3.6). Se ha intentado instalar una versión más reciente pero solo existen versiones más recientes del paquete puppet-agent. Por este motivo la puesta en marcha del servicio zabbix se ha automatizado con un script bash en vez de con puppet.

ANEXO

VERIFICACIÓN

1) Configuración de red:

- Primero se ha verificado que desde los servidores y clientes hay conectividad con el servidor ntp de la subred vlan 1ff. De esta forma se comprueba que los paquetes de los servidores y clientes salen por la vlan correspondiente (vlan 111 o vlan 112).
Para ello se ha ejecutado el comando traceroute en cada uno de ellos.
Máquina ipa1:

```
[a755232@ipa1 ~]$ traceroute6 ntp1.1.ff.es.eu.org
traceroute to ntp1.1.ff.es.eu.org (2001:470:736b:1ff::2), 30 hops max, 80 byte packets
 1  gateway (2001:470:736b:121::1)  0.457 ms  0.441 ms  0.429 ms
 2  2001:470:736b:120::1 (2001:470:736b:120::1)  0.675 ms  0.668 ms  0.688 ms
 3  ntp1.1.ff.es.eu.org (2001:470:736b:1ff::2)  1.189 ms  1.171 ms  1.160 ms
```

Máquina ipa2:

```
[a755232@ipa2 ~]$ traceroute6 ntp1.1.ff.es.eu.org
traceroute to ntp1.1.ff.es.eu.org (2001:470:736b:1ff::2), 30 hops max, 80 byte packets
 1  gateway (2001:470:736b:121::1)  0.403 ms  0.388 ms  0.377 ms
 2  2001:470:736b:120::1 (2001:470:736b:120::1)  0.468 ms  0.460 ms  0.451 ms
 3  ntp1.1.ff.es.eu.org (2001:470:736b:1ff::2)  1.065 ms  1.059 ms  1.046 ms
```

Máquina nfs1:

```
[a755232@nfs1 ~]$ traceroute6 ntp1.1.ff.es.eu.org
traceroute to ntp1.1.ff.es.eu.org (2001:470:736b:1ff::2), 30 hops max, 80 byte packets
 1  gateway (2001:470:736b:121::1)  0.271 ms  0.224 ms  0.212 ms
 2  2001:470:736b:120::1 (2001:470:736b:120::1)  0.585 ms  0.583 ms  0.573 ms
 3  ntp1.1.ff.es.eu.org (2001:470:736b:1ff::2)  1.597 ms  1.592 ms  1.587 ms
```

Máquina zabbix1:

```
[a755232@zabbix1 ~]$ traceroute6 ntp1.1.ff.es.eu.org
traceroute to ntp1.1.ff.es.eu.org (2001:470:736b:1ff::2), 30 hops max, 80 byte packets
 1  gateway (2001:470:736b:121::1)  0.363 ms  0.350 ms  0.339 ms
 2  2001:470:736b:120::1 (2001:470:736b:120::1)  0.738 ms  0.735 ms  0.727 ms
 3  ntp1.1.ff.es.eu.org (2001:470:736b:1ff::2)  1.143 ms  1.136 ms  1.127 ms
```

Máquina cliente1:

```
[root@cliente1 ~]# traceroute6 ntp1.1.ff.es.eu.org
traceroute to ntp1.1.ff.es.eu.org (2001:470:736b:1ff::2), 30 hops max, 80 byte packets
 1  2001:470:736b:122::1 (2001:470:736b:122::1)  0.303 ms  0.288 ms  0.276 ms
 2  2001:470:736b:120::1 (2001:470:736b:120::1)  0.771 ms  0.769 ms  0.759 ms
 3  ntp1.1.ff.es.eu.org (2001:470:736b:1ff::2)  0.957 ms  0.951 ms  0.942 ms
```

Máquina cliente2:

```
[root@cliente2 ~]# traceroute6 ntp1.1.ff.es.eu.org
traceroute to ntp1.1.ff.es.eu.org (2001:470:736b:1ff::2), 30 hops max, 80 byte packets
 1  2001:470:736b:122::1 (2001:470:736b:122::1)  0.228 ms  0.210 ms  0.201 ms
 2  2001:470:736b:120::1 (2001:470:736b:120::1)  0.863 ms  0.857 ms  0.848 ms
 3  ntp1.1.ff.es.eu.org (2001:470:736b:1ff::2)  1.117 ms  1.110 ms  1.100 ms
```

2) resolución DNS:

Se ha comprobado con el comando dig +trace que se accede correctamente al dominio 1.1.1.ff.es.eu.org y se resuelve la petición. Se muestra el comando dig +short para cada una de las máquinas.

Máquina ipa1:

Resolución directa

```
dig ipa1.2.1.ff.es.eu.org AAAA @8.8.8.8 +short
2001:470:736b:121::2
```

Resolución inversa

```
dig -x 2001:470:736b:121::2 @8.8.8.8 +short
ipa1.2.1.ff.es.eu.org.
```

Máquina ipa2:

Resolución directa

```
dig ipa2.2.1.ff.es.eu.org AAAA @8.8.8.8 +short
2001:470:736b:121::3
```

Resolución inversa

```
dig -x 2001:470:736b:121::3 @8.8.8.8 +short
ipa2.2.1.ff.es.eu.org.
```

Máquina nfs1:

Resolución directa

```
dig nfs1.2.1.ff.es.eu.org AAAA @8.8.8.8 +short
2001:470:736b:121::4
```

Resolución inversa

```
dig -x 2001:470:736b:121::4 @8.8.8.8 +short
nfs1.2.1.ff.es.eu.org.
```

Máquina zabbix1:

Resolución directa

```
dig zabbix1.2.1.ff.es.eu.org AAAA @8.8.8.8 +short
2001:470:736b:121::5
```

Resolución inversa

```
dig -x 2001:470:736b:121::5 @8.8.8.8 +short
zabbix1.2.1.ff.es.eu.org.
```

Máquina cliente1:

Resolución directa

```
dig cliente1.2.1.ff.es.eu.org AAAA @8.8.8.8 +short
2001:470:736b:122:5054:ff:fe01:2202
```

Resolución inversa

```
dig -x 2001:470:736b:122:5054:ff:fe01:2202 @8.8.8.8 +short
cliente1.2.1.ff.es.eu.org.
```

Máquina cliente2:

Resolución directa

```
dig cliente2.2.1.ff.es.eu.org AAAA @8.8.8.8 +short
2001:470:736b:122:5054:ff:fe01:2203
```

Resolución inversa

```
dig -x 2001:470:736b:122:5054:ff:fe01:2203 @8.8.8.8 +short
cliente2.2.1.ff.es.eu.org.
```

3) freeIPA:

Para comprobar que el servicio funciona correctamente se ha creado la cuenta de usuario “pepe” en la máquina ipa1:

```
[a755232@ipa1 ~]$ kinit admin
Password for admin@2.1.FF.ES.EU.ORG:
ipa user-add  pepe --password
Nombre: pepe
Apellido: pepe
Contraseña:
Ingrese Contraseña nuevamente para verificar:
----------------------------------
Ha sido agregado el usuario "pepe"
----------------------------------
```

Después se ha accedido a la cuenta desde la máquina cliente1:

```
ssh pepe@$c1222
Password:
Last login: Wed Jun 17 19:17:41 2020 from 2001:0:53aa:64c:2088:2fd6:acdc:71c3
-sh-4.2$ ls
cuenta_usuario_pepe.txt
```

Finalmente se ha probado a cambiar la contraseña del usuario pepe:

```
-sh-4.2$ passwd
```

```
Cambiando la contraseña del usuario pepe.
Contraseña actual:
Nueva contraseña:
Vuelva a escribir la nueva contraseña:
passwd: todos los símbolos de autenticación se actualizaron con éxito.
```

Para comprobar que la réplica freeIPA funciona correctamente se ha apagado el maestro freeIPA y se ha comprobado que la réplica ipa2 se convierte en un nuevo maestro sin que ocurra ningún problema en el sistema

4) NFSv4:

Se ha ejecutado exportfs -v en el servidor nfs1.2.1.ff.es.eu.org para comprobar que se exporta el directorio correctamente.

```
/srv/nfs4/home
2001:470:736b:122::0/64(sync,wdelay,hide,no_subtree_check,fsid=0,sec=krb5,rw,secure
,root_squash,no_all_squash)
```

Se ha ejecutado el comando showmount -e nfs1.2.1.ff.es.eu.org desde cliente1 y cliente2 para comprobar que tienen acceso a la exportación:

```
Export list for nfs1.2.1.ff.es.eu.org:
/srv/nfs4/home 2001:470:736b:122::0/64
```

Se ha comprobado al acceder a una cuenta freeIPA de usuario “Carlos” que su directorio de usuario se automonta satisfactoriamente:

```
sh-4.2$ df -h -T
S.ficheros                    Tipo    Tamaño Usados  Disp Uso% Montado en
/dev/vda2                     xfs     3,0G   1,5G  1,6G  48% /
devtmpfs                      devtmpfs   911M        0  911M   0% /dev
tmpfs                         tmpfs   920M    0  920M   0% /dev/shm
tmpfs                         tmpfs   920M   8,4M  912M   1% /run
tmpfs                         tmpfs   920M    0  920M   0% /sys/fs/cgroup
nfs1.2.1.ff.es.eu.org:/pepe nfs4      3,0G   1,5G  1,6G  48% /home/pepe
```

CÓDIGO USADO

HERRAMIENTA U

Se ha añadido la siguiente función a la herramienta u para que sea capaz de instalar módulos puppet remotamente:

```
#Instala un modulo puppet del directorio <d_despliegue> en <maquinas>
def modulo_puppet(maquinas, d_despliegue, modulo)
     f = File.expand_path(d_despliegue) # Expandir ~
     if File.file?(f + "/" +  modulo)
        begin
        maquinas.each do |host|
         Net::SSH.start(host, "a755232",:timeout=> 30) do |ssh|
```

```
                    ssh.exec!("mkdir /tmp/.puppet")
                    scp = Net::SCP.start(host, "a755232")
                    scp.upload! f + "/" + modulo, "/tmp/.puppet"
                     os = ssh.exec!("uname")
                     if os.strip == "OpenBSD" #host tiene sistema operativo openBSD
                            res = ssh.exec!("doas puppet module install
                           /tmp/.puppet/" + modulo + " --force")
                     else #host tiene sistema operativo ubuntu/centOS
                           res = ssh.exec!("sudo puppet module install
                           /tmp/.puppet/" + modulo + " --force")
                     end
                    ssh.exec!("rm -rf /tmp/.puppet")
                    puts "máquina " + host + ": exito"
                    puts res
                end

        end
        rescue
        puts "máquina : UNREACHABLE\n\n"
              end
     else
       abort "Modulo " + modulo + " no existe"
     end

end
```

DESPLIEGUE DE RED

```
#!/bin/bash
################################################################################
# Autor: Diego Marco Beisty, 755232
# Fichero: despliegue_red.sh
# Fecha:
################################################################################


# Inicialmente las máquinas disponen de una ip dinámica y puppet instalado para
# poder ser operables desde internet con la herramienta u

#######
# RED #
#######

#Testeo la conectividad con las máquinas
./u.rb maquinas_red_estado_inicial p
if [ $? -eq 1 ];then
    echo "Abortando despliegue..."
    exit 1
fi
#Instalo módulo puppet de red en todas las máquinas
```

```
./u.rb maquinas_red_estado_inicial m red.tar.gz
echo -e "-----Módulo RED instalado\n\n"

# Parámetros específicos de red máquina orouter12
./u.rb 2001:470:736b:f000:9cde:94ae:3860:2936 c router_red.pp
ssh 2001:470:736b:f000:9cde:94ae:3860:2936 'doas sh /etc/netstart'
echo -e "-----RED orouter12 OK\n\n"

# Parámetros específicos de red máquina c1212 (ipa1)
./u.rb  2001:470:736b:f000:5054:ff:fe01:2102 c ipa1_red.pp
echo -e "-----RED ipa1 OK\n\n"

# Parámetros específicos de red máquina c1213 (ipa2)
./u.rb  2001:470:736b:f000:5054:ff:fe01:2103 c ipa2_red.pp
echo -e "-----RED ipa2 OK\n\n"

# Parámetros específicos de red máquina c1214 (nfs1)
./u.rb  2001:470:736b:f000:5054:ff:fe01:2104 c nfs1_red.pp
echo -e "-----RED nfs1 OK\n\n"

# Parámetros específicos de red máquina c1215 (zabbix1)
./u.rb  2001:470:736b:f000:5054:ff:fe01:2105 c zabbix1_red.pp
echo -e "-----RED zabbix1 OK\n\n"

# Parámetros específicos de red máquina c1222 (cliente1)
./u.rb  2001:470:736b:f000:5054:ff:fe01:2202 c cliente1_red.pp
echo -e "-----RED cliente1 OK\n\n"

# Parámetros específicos de red máquina c1223 (cliente2)
./u.rb  2001:470:736b:f000:5054:ff:fe01:2203 c cliente2_red.pp
echo -e "-----RED cliente2 OK\n\n"
```

DESPLIEGUE DE NFS

```
#!/bin/bash
################################################################################
# Autor: Diego Marco Beisty, 755232
# Fichero: despliegue_nfs.sh
# Fecha:
################################################################################


#########
# NFSV4 #
#########

#Instalo módulo puppet de nfs en sevidor y clientes nfs
./u.rb nfs m nfs.tar.gz
echo -e "-----Módulo nfs instalado\n\n"

# Parámetros específicos servidor nfs1
./u.rb 2001:470:736b:121::4 c nfs1_nfs.pp
```

```
echo -e "-----servidor nfsv4 OK\n\n"

# Parámetros específicos nfs clientes
./u.rb 2001:470:736b:122:5054:ff:fe01:2202 c cliente1_nfs.pp
./u.rb 2001:470:736b:122:5054:ff:fe01:2203 c cliente2_nfs.pp
echo -e "-----clientes nfsv4 OK\n\n"
```

DESPLIEGUE DE ZABBIX

```
#!/bin/bash
################################################################################
# Autor: Diego Marco Beisty, 755232
# Fichero: despliegue_zabbix.sh
# Fecha:
################################################################################


#########
# zabbix #
#########

ssh $c1215 "sudo yum -y install mariadb-server"
ssh $c1215 "sudo yum -y install php"
ssh $c1215 "sudo systemctl start mariadb"
ssh $c1215 "sudo systemctl enable mariadb"
ssh $c1215 "sudo rpm -Uvh
https://repo.zabbix.com/zabbix/4.2/rhel/7/x86_64/zabbix-release-4.2-1.el7.noarch.rpm"
ssh $c1215 "sudo yum clean all"
ssh $c1215 "sudo yum -y install zabbix-server-mysql zabbix-web-mysql"
ssh $c1215 "sudo yum -y install zabbix-agent"

ssh $c1215 "mysql --user=root <<_EOF_
UPDATE mysql.user SET Password=PASSWORD('adminope1') WHERE User='root';
DELETE FROM mysql.user WHERE User='';
DELETE FROM mysql.user WHERE User='root' AND Host NOT IN ('localhost', '127.0.0.1',
'::1');
DROP DATABASE IF EXISTS test;
DELETE FROM mysql.db WHERE Db='test' OR Db='test\\_%';
FLUSH PRIVILEGES;
_EOF_"
ssh $c1215 "echo \"create database zabbix character set utf8 collate utf8_bin;\" | mysql
-uroot -padminope1"
ssh $c1215 "echo \"grant all privileges on zabbix.* to zabbix@localhost identified by
'adminope1';\" | mysql -uroot -padminope1"
ssh $c1215 "echo \"flush privileges;\" | mysql -uroot -padminope1"
ssh $c1215 "zcat /usr/share/doc/zabbix-server-mysql*/create.sql.gz | mysql -uzabbix
zabbix -padminope1"
./u.rb 2001:470:736b:121::5 c zabbix1_zabbix.pp
ssh $c1215 "sudo systemctl restart httpd"
ssh $c1215 "sudo systemctl enable httpd"
ssh $c1215 "sudo systemctl start zabbix-server"
```

```
ssh $c1215 "sudo systemctl start zabbix-agent"
ssh $c1215 "sudo systemctl enable zabbix-server"
ssh $c1215 "sudo systemctl enable zabbix-agent"


./u.rb clientes_zabbix s "sudo rpm -Uvh
https://repo.zabbix.com/zabbix/4.2/rhel/7/x86_64/zabbix-release-4.2-1.el7.noarch.rpm"
./u.rb clientes_zabbix s "sudo yum -y clean all"
./u.rb clientes_zabbix s "sudo yum -y install zabbix-agent"
./u.rb clientes_zabbix c clientes_zabbix.pp
./u.rb clientes_zabbix s "sudo systemctl start zabbix-agent"
./u.rb clientes_zabbix s "sudo systemctl enable zabbix-agent"
```

PAQUETE PUPPET RED

init.pp

```
class red(
    $tipo_nodo,
    $ip_nodo="",
    $dns1_nodo,
    $dns2_nodo="",
    $nombre_dns_nodo,
){
    case $tipo_nodo {
       'router_openbsd': {
        class{'red::router_openbsd':
              dns1 => $dns1_nodo,
              nombre_dns => $nombre_dns_nodo,
        }
       }
       'servidor_centos': {
        class{'red::servidor_centos':
              ip=> $ip_nodo,
              dns1=> $dns1_nodo,
              dns2=> $dns2_nodo,
              nombre_dns => $nombre_dns_nodo,
        }
       }

       'cliente_centos': {
        class{'red::cliente_centos':
              dns1=> $dns1_nodo,
              dns2=> $dns2_nodo,
              nombre_dns => $nombre_dns_nodo,
        }
       }
```

```
    }
}
```

router_openbsd.pp

```
class red::router_openbsd(
    $dns1,
    $nombre_dns,
){
    file {'/etc/hostname.vio0':
               ensure=> file,
               mode=>'0644',
               content=>"-inet6
up
-autoconfprivacy
",
       }

        file{'/etc/hostname.vlan120':
               ensure=>file,
               mode=>'0644',
               content=>"inet6 alias 2001:470:736b:120::2 60 vlan 120 vlandev vio0
-autoconfprivacy
",
       }

    file{'/etc/mygate':
               ensure=>file,
               mode=>'0644',
               content=>"2001:470:736b:120::1
",
       }

    file{'/etc/myname':
               ensure=>file,
               mode=>'0644',
               content=>"${nombre_dns}
",
       }


      file{'/etc/resolv.conf':
               ensure=>file,
               mode=>'0644',
               content=>"nameserver ${dns1}
",
       }


        file{'/etc/ntpd.conf':
               ensure=>file,
               mode=>'0644',
```

```
                content=>"server ${dns1}
",
        }

    file{'/etc/hostname.vlan121':
                ensure=>file,
                mode=>'0644',
                content=>"inet6 alias 2001:470:736b:121::1 64 vlan 121 vlandev vio0
-autoconfprivacy
",
        }


    file{'/etc/hostname.vlan122':
                ensure=>file,
                mode=>'0644',
                content=>"inet6 alias 2001:470:736b:122::1 64 vlan 122 vlandev vio0
-autoconfprivacy
",
        }


    file{'/etc/rc.conf.local':
                ensure=>file,
                mode=>'0644',
                content=>"pflogd_flags=NO
sndiod_flags=NO
pf=NO
ntpd_flags=-s
rad_flags=\"\"
",
}


    file{'/etc/rad.conf':
                ensure=>file,
                mode=>'0644',
                content=>"interface vlan122
",
        }

    service { rad:
                ensure=>running,
                subscribe => File["/etc/rad.conf"],
    }

    file{'/etc/sysctl.conf':
        ensure=>file,
        mode=>'0644',
        content=>"net.inet6.ip6.forwarding=1
",
    }
    exec { "update-forwarding":
```

```
                command         => "sysctl net.inet6.ip6.forwarding=1",
                path            => "/bin:/usr/bin:/sbin:/usr/sbin",
        }
}
```

servidor_centos.pp

```
class red::servidor_centos(
    $ip,
    $dns1,
    $dns2,
    $nombre_dns,

) {
    file {'/etc/sysconfig/network-scripts/ifcfg-eth0':
                ensure => file,
                mode => '0644',
                content =>"TYPE=Ethernet
PROXY_METHOD=none
BROWSER_ONLY=no
DEFROUTE=yes
IPV4_FAILURE_FATAL=no
IPV6INIT=no
NAME=eth0
UUID=7f2813c5-c37c-4b77-ac7c-3b0dc20651e7
DEVICE=eth0
ONBOOT=yes
IPV6_PRIVACY=no
",
        }

    file {'/etc/sysconfig/network-scripts/ifcfg-eth0.121':
                ensure => file,
                mode => '0644',
                content =>"VLAN=yes
TYPE=vlan
PHYSDEV=eth0
DEVICE=eth0.121
VLAN_ID=121
GVRP=no
REORDER_HDR=yes
MVRP=no
PROXY_METHOD=none
BROWSER_ONLY=NO
IPV6INIT=yes
IPV6_AUTOCONF=no
IPV6ADDR=${ip}
IPV6_DEFAULTGW=2001:470:736B:121::1
DNS1=${dns1}
DNS2=${dns2}
DOMAIN=2.1.ff.es.eu.org
IPV6_DEFROUTE=yes
IPV6_FAILURE_FATAL=no
```

```
IPV6_ADDR_GEN_MODE=stable-privacy
NAME=eth0.121
UUID=6dededc0-e86c-4a4f-9509-5e874ff851e2
ONBOOT=yes
",
       }

    file {'/etc/sysctl.conf':
               ensure => file,
               mode => '0644',
               content =>"net.ipv6.conf.eth0.use_tempaddr=0
net.ipv6.conf.eth0.autoconf=0
net.ipv6.conf.eth0.accept_ra=0
",
       }



     file {'/etc/hostname':
               ensure => file,
               mode => '0644',
               content =>"${nombre_dns}
",
        }

    file {'/etc/hosts':
               ensure => file,
               mode => '0644',
               content =>"127.0.0.1   localhost localhost.localdomain localhost4
localhost4.localdomain4
::1            localhost localhost.localdomain localhost6 localhost6.localdomain6

${ip}  ${nombre_dns}
",
       }

    service{'network':
         ensure=>running,
         subscribe => File["/etc/sysconfig/network-scripts/ifcfg-eth0.121"],
    }
}
```

cliente_centos.pp

```
class red::cliente_centos(
    $dns1,
    $dns2,
    $nombre_dns,

) {
    file {'/etc/sysconfig/network-scripts/ifcfg-eth0':
               ensure => file,
               mode => '0644',
```

```
              content =>"TYPE=Ethernet
PROXY_METHOD=none
BROWSER_ONLY=no
DEFROUTE=yes
IPV4_FAILURE_FATAL=no
IPV6INIT=no
NAME=eth0
UUID=7f2813c5-c37c-4b77-ac7c-3b0dc20651e7
DEVICE=eth0
ONBOOT=yes
IPV6_PRIVACY=no
",
       }

    file {'/etc/sysconfig/network-scripts/ifcfg-eth0.122':
              ensure => file,
              mode => '0644',
              content =>"VLAN=yes
TYPE=vlan
PHYSDEV=eth0
DEVICE=eth0.122
VLAN_ID=122
REORDER_HDR=yes
GVRP=no
MVRP=no
PROXY_METHOD=none
BROWSER_ONLY=no
IPV6_INIT=yes
IPV6_AUTOCONF=yes
DNS1=${dns1}
DNS2=${dns2}
DOMAIN=2.1.ff.es.eu.org
IPV6_DEFROUTE=yes
IPV6_FAILURE_FATAL=no
IPV6_ADDR_GEN_MODE=stable-privacy
NAME=eth0.122
UUID=3031016a-4c31-4175-bf0d-55234790465e
ONBOOT=yes

",
       }

    file {'/etc/sysctl.conf':
              ensure => file,
              mode => '0644',
              content =>"net.ipv6.conf.eth0.use_tempaddr=0
net.ipv6.conf.eth0.autoconf=0
net.ipv6.conf.eth0.accept_ra=0
",
       }
```

```
     file {'/etc/hostname':
               ensure => file,
               mode => '0644',
               content =>"${nombre_dns}
",
         }


    service{'network':
         ensure=>running,
         subscribe => File["/etc/sysconfig/network-scripts/ifcfg-eth0.122"],
    }
}
```

PAQUETE NFS

init.pp

```
class nfs(
    $tipo_nodo,
){
    case $tipo_nodo {
        'servidor_nfs': {
         class{'nfs::servidor_nfs':}
        }
        'cliente_nfs': {
         class{'nfs::cliente_nfs':}
        }

    }
}
```

servidor_nfs.pp

```
class nfs::servidor_nfs {
    package { "nfs-utils":
                ensure => present,
         }

    service {"nfslock":
         ensure => running,
                enable => true,
                require => Package["nfs-utils"],
         }
```

```
    service { "nfs":
                ensure => running,
                enable => true,
                require => Service["nfslock"],
            }

    file {['/srv/nfs4',
                        '/srv/nfs4/home', ]:
        ensure=> directory,
    }

    file {'/etc/exports':
        ensure=>file,
        content=>"/srv/nfs4/home
2001:470:736b:122::0/64(rw,sync,fsid=0,sec=krb5,no_subtree_check)
",
        notify => Service["nfs"],
    }

}
```

cliente_nfs.pp

```
class nfs::cliente_nfs {
    package { "nfs-utils":
                ensure => present,
           }


    service { "autofs":
                ensure => running,
                enable => true,
                require => Package["nfs-utils"],
           }

            file {'/etc/auto.master':
                ensure=> file,
                content=>"
/misc   /etc/auto.misc
/net    -hosts
+dir:/etc/auto.master.d
+auto.master
/home   /etc/auto.home
",
           }

            file {'/etc/auto.home':
                ensure=>file,
                content=>"*    -rw,fstype=nfs4,sec=krb5     nfs1.2.1.ff.es.eu.org:/&
",
                notify => Service["autofs"],
           }
```

```
}
```

AÑADIR DIRECTORIO DE USUARIO

```
#!/bin/bash
# Autor: Diego Marco, 755232
# Fichero: crear_dirhome.sh
# Coms: Crea los directorios de usuarios freeIPA a exportar en el servidro
#     nfs1 del proyecto1 parte 2 de la asignatura Administracion de sistemas 2

if [ ! $# -eq 1 ];then
        echo "Usage: ./crear_dirhome.sh <nombre>"
        exit
fi

nfs1=2001:470:736b:121::4 #Servidor nfs

#configuracion servidor
uid=$(ssh $nfs1 "id -u $1" 2>/dev/null)

numero='^[0-9]+$'
if ! [[ $uid =~ $numero ]] ; then
   echo "Error: Cuenta de usuario $1 no existe"; exit 1
fi

if ssh $nfs1 "[ -d /srv/nfs4/home/$1 ]"; then
    echo "Error: Directorio de usuario $1 ya existe"; exit 1
fi

ssh $nfs1 "sudo mkdir /srv/nfs4/home/$1"
ssh $nfs1 "sudo cp -r /etc/skel/. /srv/nfs4/home/$1"
ssh $nfs1 "sudo touch /srv/nfs4/home/$1/cuenta_usuario_$1.txt"
ssh $nfs1 "sudo chown -R $uid:$uid /srv/nfs4/home/$1"
ssh $nfs1 "sudo exportfs -a"
```