

Samba LVS HOWTO

Fred Lacombe <lacombe (at) linagora (dot) com>

2004-02-18

Revision History

Revision 0.9.1 2004-02-18 L(c)
 first draft

This document aims to describe an load-balancing architecture for Samba services. All technologies and configurations are described as well as some difficulties that rise from LVS and NetBIOS name resolution.

Table of Contents

Introduction	1
Copyright and License	1
Disclaimer	1
Feedback	2
Hardware needs	2
Software configuration	3
PDC real server	3
Many PDCs servers	9
LVS Director	10
Testing the architecture	11
Limitations and improvments	11

Introduction

As network size is growing, some load-balancing solutions may be consider. But for some protocols, name resolutions problems emerge as well as full connections dialog. In the Samba particular case, it exists some solutions through DFS implementation, but it keeps a full load for authentication purpose. We intents here to present a way to have Samba PDC architecture that will be distributed through a LVS director.

As this solution has been recently tested, we can say that it works but some other problems appear. So we cannot say right now that we have an answer for a complete Samba load-balancing process. Here is the modus operandi used to get a distributed authentication and shares access. For name resolution and browsing process, we still have to investigate, some ideas are to be tested.

Copyright and License

This document, *Samba LVS HOWTO*, is copyrighted (c) 2004 by *Fred Lacombe*. Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.1 or any later version published by the Free Software Foundation; with no Invariant Sections, with no Front-Cover Texts, and with no Back-Cover Texts. A copy of the license is available at <http://www.gnu.org/copyleft/fdl.html> [<http://www.gnu.org/copyleft/fdl.html>].

Linux is a registered trademark of Linus Torvalds.

Disclaimer

--4pc --4pc

No liability for the contents of this document can be accepted. Use the concepts, examples and information at your own risk. There may be errors and inaccuracies, that could be damaging to your system. Proceed with caution, and although this is highly unlikely, the author(s) do not take any responsibility.

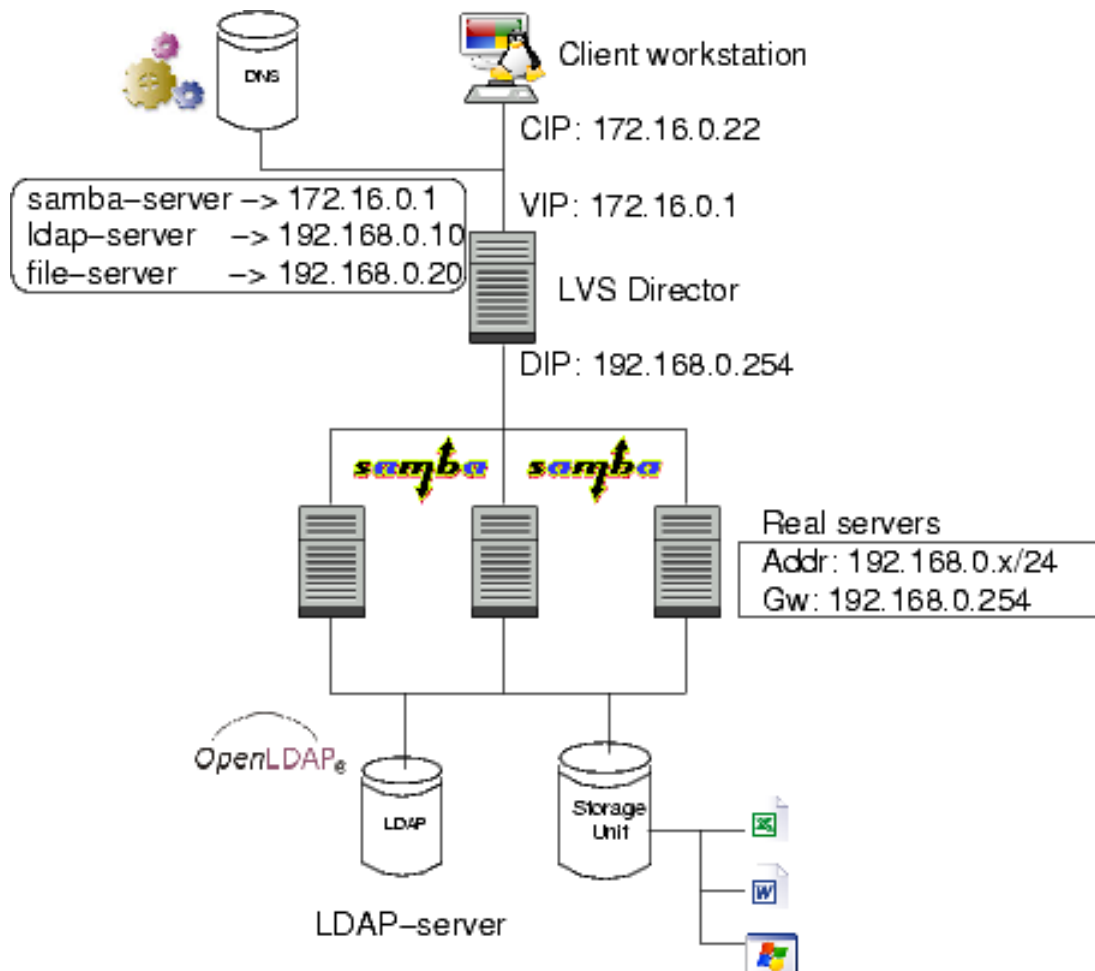
All copyrights are held by their by their respective owners, unless specifically noted otherwise. Use of a term in this document should not be regarded as affecting the validity of any trademark or service mark. Naming of particular products or brands should not be seen as endorsements.

Feedback

Feedback is most certainly welcome for this document. Send your additions, comments and criticisms to the following email address : <lacombe (at) linagora (dot) com>.

Hardware needs

The current architecture needs ideally four machines in order to implement an LVS director, at least two real servers and a client for testing the query repartition.



Load-balancing in a Samba architecture

In this scheme, the front-end machine will be called director, as commonly used in LVS description. The real servers are configured in a mirror way, they offer a strictly identical service, in our study case a Samba PDC configuration.

- -4pc - -4pc

The following picture describes the global network architecture as well as the names and addresses in use for each computer.

Software configuration

Here is a list of all software we need to see to deploy such an architecture. They will be shown respectively to the methodology we will apply in this document.

- Samba
- LDAP
- Iptables
- Routing
- IPVS / ipvsadm
- DNS

PDC real server

In our study case, we focus on a PDC server for authentication purpose with an underlying LDAP directory. We have also to define a share in order to validate both authentication and access to resource from a client.

Samba service

First, we want to define a real server configuration in order to get an operating samba server. We want to build a PDC configuration. This how-to is not designed to describe Samba capabilities but I will give as many details as I can to point out some of the problems that I have encountered. I have built this configuration using samba-3.0.0 on a Red Hat AS 3. I believe that this methodology should apply to any distribution.

All the following occurs in the `/etc/samba/smb.conf` file for Samba configuration. First, we have to define the NetBIOS domain name and server name. Our PDC should answer to a generic name that is shared between all real servers. To keep an individual identification for each server, we can use the `netbios aliases` parameter to have many names for a server.

```
[global]
workgroup = DOMAIN
netbios name = server1
netbios aliases = samba-server
```

Next, we have to define the server as a PDC, the following parameters force the Samba server to become a domain controller.

```
security = user
encrypt passwords = Yes

local master = Yes
domain master = Yes
domain logons = Yes
preferred master = Yes
```

```
--4pc --4pc
os level = 64
```

Finally, we have to define the LDAP support. The choice of LDAP is quite important in such an architecture, we want to share the load of Samba service but keep a single point for authentication database. As a first scheme, we will consider a single directory service, but for performance reasons, we can deploy local LDAP replicas on each Samba servers. This point will be focused later on. Here is the LDAP parameters needed for Samba to rely on:

```
passdb backend = ldapsam:ldap://ldap-server

ldap suffix = dc=domain
ldap admin dn = "cn=admin,dc=domain"
ldap filter = "(&(uid=%u)(objectclass=sambaSamAccount))"
ldap user suffix = ou=Users
ldap group suffix = ou=Groups
ldap machine suffix = ou=Computers
```

LDAP service

As defined in the previous configuration file, a corresponding LDAP directory must be accessible through the server `ldap-server`. This directory service has to integrate the Samba schema that is joined in the Samba archive. We have to copy the `samba.schema` in the `/etc/ldap/schema` directory and to configure the `/etc/ldap/slapd.conf` file. This file should be similar to this:

```
# Schema and objectClass definitions
include /etc/ldap/schema/core.schema
include /etc/ldap/schema/cosine.schema
include /etc/ldap/schema/nis.schema
include /etc/ldap/schema/inetorgperson.schema
include /etc/ldap/schema/samba.schema

# Schema check allows for forcing entries to
# match schemas for their objectClasses's
schemacheck on

# Where the pid file is put. The init.d script
# will not stop the server if you change this.
pidfile /var/run/slapd/slapd.pid

# List of arguments that were passed to the server
argsfile /var/run/slapd.args

# Read slapd.conf(5) for possible values
loglevel 0

# Where the dynamically loaded modules are stored
modulepath /usr/lib/ldap
moduleload back_bdb

#####
# Specific Backend Directives for bdb:
backend bdb

#####
```

```
- -4pc - -4pc
# Specific Directives for database #1, of type bdb:
database bdb

# The base of your directory in database #1
suffix "dc=domain"

# Where the database file are physically stored for database #1
directory "/var/lib/ldap"

# Administrator ID
rootdn "cn=admin,dc=domain"

# Administrator password : secret
rootpw {SSHA}T3z64Tw1J+AOQ/dli1GKl4kngh2gH8jh
```

For optimization purpose and to secure passwords access, some more parameters may be needed such as those :

```
# Indexing options for database #1
index objectClass eq

# Samba indexes
index cn,sn,uid,displayName pres,sub,eq
index uidNumber,gidNumber eq
index sambaSID eq
index sambaPrimaryGroupSID eq
index sambaDomainName pres,eq
index default sub

# Save the time that the entry gets modified, for database #1
lastmod on

# Where to store the replica logs for database #1
# relogfile /var/lib/ldap/relog

# These access lines apply to database #1 only
access to attribute=userPassword,sambaNTPassword,sambaLMPassword
by dn="cn=admin,dc=domain" write
by anonymous auth
by self write
by * none

# The admin dn has full write access
access to *
by * read
```

We have now to give Samba the password for admin dn entry, this must be done with the following command: `smbpasswd -w secret`, and it will create a `secret.tdb` file that will keep the LDAP password protected.

To deploy the LDAP database, we can just need a standard tree separating users, group and computers accounts. Both solutions can be considered here, either use a LDIF file to build LDAP structure by hand or use an administration tool such as LDAP Account Manager (LAM) which create each branch of the tree for us. I will give now the LDIF result for the database we consider.

```
- -4pc - -4pc

# domain
dn: dc=domain
objectClass: organization
objectClass: dcObject
dc: domain
o: domain

# Users, domain
dn: ou=Users,dc=domain
objectClass: organizationalunit
ou: Users

# Groups, domain
dn: ou=Groups,dc=domain
objectClass: organizationalunit
ou: Groups

# Computers, domain
dn: ou=Computers,dc=domain
objectClass: organizationalunit
ou: Computers

# Domains, domain
dn: ou=Domains,dc=domain
objectClass: organizationalunit
ou: Domains
```

To complete the configuration to create a NetBIOS domain name to our PDC, we choose `example` in our case and we add the Windows administrative groups: the Domain Admins, Domain Users and Domain Guests groups.

```
# example, Domains, domain
dn: sambaDomainName=example,ou=Domains,dc=domain
objectClass: sambaDomain
sambaDomainName: example
sambaSID: S-1-5-21-0
sambaAlgorithmicRidBase: 1000

# ntadmins, Groups, domain
dn: cn=ntadmins,ou=Groups,dc=domain
objectClass: posixGroup
objectClass: sambaGroupMapping
cn: ntadmins
gidNumber: 10000
description: Domain Admins
sambaSID: S-1-5-21-0-512
sambaGroupType: 2
displayName: Domain Admins

# ntusers, Groups, domain
dn: cn=ntusers,ou=Groups,dc=domain
objectClass: posixGroup
objectClass: sambaGroupMapping
cn: ntusers
gidNumber: 10001
description: Domain Users
```

```
- -4pc - -4pc
sambaSID: S-1-5-21-0-513
sambaGroupType: 2
displayName: Domain Users

# ntguests, Groups, domain
dn: cn=ntguests,ou=Groups,dc=domain
objectClass: posixGroup
objectClass: sambaGroupMapping
cn: ntguests
gidNumber: 10002
description: Domain Guests
sambaSID: S-1-5-21-0-514
sambaGroupType: 2
displayName: Domain Guests
```

Finally here is an example of a user declaration inside our LDAP tree. Actually, this user is a generic definition that we can reuse in scripts for generating a large number of users. Do not forget that we may want to study a load-balancing solution and those users may simulate later on a real activity on a large scale network.

```
# template, Users, domain
dn: uid=template,ou=Users,dc=domain
objectClass: posixAccount
objectClass: shadowAccount
objectClass: inetOrgPerson
objectClass: sambaSamAccount
cn: template
uid: template
uidNumber: 10000
gidNumber: 10001
homeDirectory: /home/template
givenName: template
sn: template
loginShell: /bin/bash
gecos: template template
description: template template
displayName: template
userPassword:: e1NTSEF9Sng3WUpNazdsdVZsU0E2aHYvN3JlS085UVdBaHZFblU=
shadowMin: 1
shadowMax: 365
shadowWarning: 10
shadowInactive: 10
shadowLastChange: 12451
shadowExpire: 21914
sambaSID: S-1-5-21-0-21000
sambaPrimaryGroupSID: S-1-5-21-0-513
sambaAcctFlags: [UX      ]
sambaHomePath: \\samba-server\template
sambaHomeDrive: U:
sambaDomainName: example
sambaNTPassword: 6D3986E540A63647454A50E26477EF94
sambaLMPassword: 9D51F8EC4F16C9ADAAD3B435B51404EE
sambaPwdLastSet: 1075829073
sambaPwdCanChange: 1041375601
sambaPwdMustChange: 1893452401
```

--4pc - -4pc

The definition is certainly too complete for our use as it supports POSIX authentication parameters. We can shorten this file by suppressing shadow account values. You may notice as well that the home directory of the user is referred to samba-server. This point will be focus later, if we intent to centralize all data for storage solutions and concurrent accesses.

Share definition

Now that we have PDC configuration and a minimum of one user defined on it, we may need to create Samba share to validate connection to our server. Let notice that there is already the home directory of our user that is accessible, here we want to create a share for anonymous usage, like a /tmp area to point out the alternative connection from one server to the other.

We must complete the /etc/samba/smb.conf file with the necessary shares. First of all, on a PDC we have to define a [netlogon] section as a logon scripts repository, as well as a [profiles] for users profiles hosting.

```
[netlogon]
path = /var/lib/samba/netlogon
writable = No
browsable = No
locking = No
```

```
[profiles]
path = /var/lib/samba/profiles
writable = Yes
browsable = No
guest ok = Yes
create mask = 0600
directory mask = 0700
```

Next, we have to share the users home directory and our temporary share. Those definitions do not intend to be a secure configuration, just to offer an access point to files one a PDC server.

```
[homes]
comment = %u's Home Directory
writable = Yes
browsable = No
guest ok = No
create mask = 0600
directory mask = 0700
[test]
comment = Temporary test directory
path = /tmp
browsable = Yes
writable = Yes
guest ok = Yes
```

Samba tests

Before any other step, we have to test our PDC configuration. We must validate the following points to be sure of a normal behavior till now:

--4pc - -4pc

- List all browseable shares
- Authenticate a user against Samba/LDAP
- Access to a user's home directory
- Access to the test share
- Write in the test share to have witness file

To realize those tests we rely on the samba tools suite, `smbclient` should be enough to pass all these points. Do not forget to check the Samba syntax with `testparm` to raise some miswriting errors. Once it is passed, we can start the interesting part of our configuration.

Many PDCs servers

After we verify the Samba configuration and that the service is running, we try to authenticate against the LDAP database through Samba. If the challenge succeeds, the Samba configuration can be duplicated on the other real server. We have only one parameter to change on our new real server, it the `netbios` name parameter to keep an individual naming for each of our servers. If you rely on DNS resolution or local hostname usage, you don't even need to change this value.

Since we have configured two similar PDC with the same name and domain, we need to isolate them, to get them blind in the NetBIOS context. For that goal, some `iptables` rules can jail each of the PDCs. In that way they can coexist on the same network and provide the same service as they ignore each other presence.

Iptables jails

The only machine that should connect to the real servers is the director. The following rules, even if they are too broad, are designed to this goal. Theoretically, we could restrict the rules to hide PDC presence by forbidding only 139 and 445 TCP ports as well as 137 and 138 UDP ports between real servers.

```
DIRECTOR=192.168.0.254
LOCAL=192.168.0.1

# Default policy
iptables -P INPUT DROP
iptables -P OUTPUT DROP
iptables -P FORWARD DROP

# Loopback rules: accept everything
iptables -A INPUT -i lo -j ACCEPT
iptables -A OUTPUT -o lo -j ACCEPT

# Rules definitions
iptables -N in
iptables -N out
iptables -A INPUT -i eth0 -j in
iptables -A OUTPUT -o eth0 -j out

# Blind config
# We accept all connections from the director
iptables -A in -s $DIRECTOR -j ACCEPT
iptables -A out -d $DIRECTOR -j ACCEPT
# We accept local connection on IP address
iptables -A in -s $LOCAL -j ACCEPT
```

```
--4pc --4pc
iptables -A out -d $LOCAL -j ACCEPT
```

This firewall configuration must be deployed on both real servers to keep a mirror configuration. This kind of policy should be the most adaptive if we consider many Samba real servers, we only open allowed connections instead of forbidding a vertex of Samba server nodes. As an example, we can consider that we first rely on a single LDAP service located on `ldap-server`. To authorize all dialogs between Samba and LDAP, we need to allow some more flexibility adding those definitions to the previous declaration:

```
LDAP=192.168.0.10

# Samba/LDAP dialog
iptables -A in -p tcp -s $LDAP --sport 389 -j ACCEPT
iptables -A out -p tcp -d $LDAP --dport 389 -j ACCEPT
```

Packet routing

As we have now many real servers, we need to consider them on a dedicated sub-network that should be only accessible from the LVS director. With that point of view, we must force the reverse routing process through the director. That means the director is defined as a gateway for the real servers for each Samba servers.

So we have to be careful about route table definition. We must check and set it right with `ip` or `route` functions to be conform to the following description:

```
# route
Destination      Gateway          Genmask          Indic Metric Ref       Use Iface
192.168.0.0      *                255.255.255.0   U      0      0        0 eth0
default          192.168.0.254   0.0.0.0         UG     0      0        0 eth0
```

LVS Director

Once the real servers are well configured, jailed and routed, we have now to give the correct parameters to the LVS director. In order to realize this operation, we need a kernel that has IPVS capabilities. The administration tool via command line, `ipvsadm` is also required. Focus is given on the version of the patch and `ipvsadm` command that must be coherent the one for the other in order to work. A complete description of kernel patch and corresponding API is describe on LVS web site.

LVS rules are quite similar to `iptables` ones:

```
# Virtual interface
VIP=172.16.0.1

# Real servers interfaces
REAL1=192.168.0.1
REAL2=192.168.0.2

# Virtual service declaration
ipvsadm -A -t $VIP:445 -s rr
ipvsadm -A -t $VIP:139 -s rr
ipvsadm -A -u $VIP:137 -s rr
```

```
- -4pc - -4pc
ipvsadm -A -u $VIP:138 -s rr

# Real servers assignation
ipvsadm -a -t $VIP:445 -r $REAL1:445 -m -w 1
ipvsadm -a -t $VIP:445 -r $REAL2:445 -m -w 1
ipvsadm -a -t $VIP:139 -r $REAL1:139 -m -w 1
ipvsadm -a -t $VIP:139 -r $REAL2:139 -m -w 1
ipvsadm -a -u $VIP:137 -r $REAL1:137 -m -w 1
ipvsadm -a -u $VIP:137 -r $REAL2:137 -m -w 1
ipvsadm -a -u $VIP:138 -r $REAL1:138 -m -w 1
ipvsadm -a -u $VIP:138 -r $REAL2:138 -m -w 1
```

Kernel options and routing

It is compulsory to activate the IP forwarding option in the kernel to have a correct routing scheme. The following options must be set in the file `/etc/sysctl.conf` for routing purpose and to avoid route redefinition:

```
net/ipv4/conf/all/ip_forward = 1
net/ipv4/conf/all/accept_redirects = 0
net/ipv4/conf/all/send_redirects = 0
```

Testing the architecture

Last but not least, we can now test our configuration. As we need to connect to a computer named `samba-server`, an association must be set to map the virtual IP, which is the only access point to Samba server. This name resolution has to be defined in a DNS server or an external WINS service. For testing only, you can also put it temporarily in your `/etc/hosts`.

It is time to connect to the load balancing mechanism. A Samba request to access the share on one of the PDC servers must be done, with `smbmount` or `smbclient`:

```
smbmount //samba-server/test -o username=<user-id>
```

With an audit tool we can monitor the flow through the director and on each of the real servers we can run `smbstatus` command to check the open shares are alive connections. When we retry the previous command, then we can point out that the connection is routed to the second real servers.

Limitations and improvements

The previous architecture still suffers from drawbacks that we have to focus on. As a non-exhaustive list, we may notice:

- What about LVS-DR and LVS-Tun ?
- Data synchronisation
- NetBIOS name resolution
- Browsing on such an architecture