



東南大學
SOUTHEAST UNIVERSITY

云计算技术及应用 课程实践报告

标 题	云存储技术实践
学 号	232349
姓 名	李志成

东南大学计算机科学与工程学院

二零二四年四月

目录

实验三 云存储技术实践.....	1
1 实验目标与要求	1
2 实验一：搭建 GlusterFS	1
2.1 下载并准备 Fedora 镜像	1
2.2 安装虚拟机	1
2.3 固定虚拟机 IP 地址	3
2.4 格式化、挂载分区（bricks）	4
2.5 安装 GlusterFS	4
2.6 配置可信任池	5
2.7 建立 GlusterFS 卷	6
2.8 测试 GlusterFS 卷	7
3 实验二：搭建 ownCloud	8
3.1 准备工作	8
3.1.1 关闭防火墙	8
3.1.2 关闭 selinux	8
3.1.3 更新 yum 源	9
3.2 安装 ownCloud	9
3.2.1 安装 samba、httpd	9
3.2.2 安装 php7.4	9
3.2.3 安装 mariadb	10
3.2.4 设置 samba 开机启动	10
3.2.5 设置 httpd 开机启动	11
3.2.6 设置 mariadb 开机启动	11
3.2.7 修改 /var/www/html 权限	11
3.2.8 安装 ownCloud	11
3.2.9 重启 httpd	12
3.3 注册测试	12
4 思考并拓展实践	13

1 实验目标与要求

- 1) 在 VMware 中，安装 3 个 Fedora 虚拟机。
- 2) 分别在每个虚拟机节点上部署 Fedora。
- 3) 在 Linux 系统上实现 Native 挂载；实验验证 GlusterFS 集群。
- 4) 在其中一台服务器上搭建 ownCloud，创建自己的私人云盘系统。
- 5) 思考如何结合 GlusterFS 七种卷的特性，为 ownCloud 用户存储的文档提供高可靠和高可用的服务。

2 实验一：搭建 GlusterFS

2.1 下载并准备 Fedora 镜像

https://download.fedoraproject.org/pub/fedora/linux/releases/38/Server/x86_64/iso/Fedora-Server-dvd-x86_64-38-1.6.iso

浏览器输入上述链接，下载 Fedora-server 38 镜像。

2.2 安装虚拟机

硬件配置：

虚拟机内存：2048MB

虚拟磁盘大小：20GB

注意事项：

1. 操作系统选择 “Fedora 64 位”



2. 虚拟磁盘类型选择“SCSI(S)”



点击“编辑虚拟机设置”，编辑刚刚创建的虚拟机。手动为每个虚拟机新增一个虚拟磁盘，大小仍为 20GB，虚拟磁盘类型仍选择“SCSI(S)”。



通过命令“fdisk -l”查看：

```
l1zc@192 ~]$ su
Password:
[root@192 l1zc]# fdisk -l
Disk /dev/sda: 20 GiB, 21474836480 bytes, 41943040 sectors
Disk model: VMware Virtual S
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disklabel type: gpt
Disk identifier: 6241F75A-6A13-4F15-98EF-26678A22C329

Device            Start      End  Sectors  Size Type
/dev/sda1         2048      4095    2048    1M BIOS boot
/dev/sda2         4096    2101247  2097152    1G Linux filesystem
/dev/sda3        2101248  41940991 39839744    19G Linux LVM

Disk /dev/sdb: 20 GiB, 21474836480 bytes, 41943040 sectors
Disk model: VMware Virtual S
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes

Disk /dev/mapper/fedora_192-root: 15 GiB, 16106127360 bytes, 31457280 sectors
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes

Disk /dev/zram0: 1.91 GiB, 2047868928 bytes, 499968 sectors
Units: sectors of 1 * 4096 = 4096 bytes
Sector size (logical/physical): 4096 bytes / 4096 bytes
I/O size (minimum/optimal): 4096 bytes / 4096 bytes
[root@192 l1zc]#
```

可以看到，此时虚拟机内新增了一块磁盘 /dev/sdb。

2.3 固定虚拟机 IP 地址

使用 ifconfig 命令，查看当前网络基本信息：

```
[root@192 l1zc]# ifconfig
ens160: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 192.168.61.130 netmask 255.255.255.0 broadcast 192.168.61.255
    inet6 fe80::20c:29ff:fe2f:58ad prefixlen 64 scopeid 0x20<link>
    ether 00:0c:29:2f:58:ad txqueuelen 1000 (Ethernet)
    RX packets 75 bytes 7507 (7.3 KiB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 109 bytes 9199 (8.9 KiB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536
    inet 127.0.0.1 netmask 255.0.0.0
    inet6 ::1 prefixlen 128 scopeid 0x10<host>
    loop txqueuelen 1000 (Local Loopback)
    RX packets 0 bytes 0 (0.0 B)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 0 bytes 0 (0.0 B)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
```

当前网卡名称为“ens160”，IP 地址为 192.168.61.130，子网掩码为 255.255.255.0（等价于 /24）。

使用 ip route show 命令，查看当前网关信息：

```
[root@192 l1zc]# ip route show
default via 192.168.61.2 dev ens160 proto dhcp src 192.168.61.130 metric 100
192.168.61.0/24 dev ens160 proto kernel scope link src 192.168.61.130 metric 100
```

当前网关地址为 192.168.61.2。

使用 nmcli 命令进行网卡配置，修改 ifcfg-ens160，以固定 IP：

```
nmcli connection modify ens160 ipv4.addresses 192.168.61.130/24 ipv4.gateway 192.168.61.2
ipv4.dns 8.8.8.8 ipv4.method manual connection.autoconnect yes
```

```
#    ipv4.method manual          - - - IPv4 模式（auto=自动 manual=静态）
#    ipv4.addresses 192.168.61.130/24 - - - IP 地址
#    ipv4.gateway 192.168.61.2   - - - 网关
#    ipv4.dns 8.8.8.8            - - - dns 解析
#    connection.autoconnect yes  - - - 开机自动连接

nmcli connection up ens160

#    让更新的配置生效
```

```
(root@192 lzc) # nmcli connection modify ens160 ipv4.addresses 192.168.61.130/24 ipv4.gateway 192.168.61.2 ipv4.dns 8.8.8.8 ipv4.method manual connection.autoconnect yes
(root@192 lzc) # nmcli connection up ens160
Connection successfully activated (D-Bus active path: /org/freedesktop/NetworkManager/ActiveConnection/3)
```

终端显示，已经成功让更新的配置生效。

2.4 格式化、挂载分区（bricks）

在每个虚拟机节点上执行以下命令：

```
mkfs.xfs -i size=512 /dev/sdb
mkdir -p /data/brick1
echo '/dev/sdb /data/brick1 xfs defaults 1 2' >> /etc/fstab
mount -a && mount
```

```
(root@192 lzc) # mkfs.xfs -i size=512 /dev/sdb
meta-data=/dev/sdb            isize=512    agcount=4, agsize=1310720 blks
=                               sectsz=512   attr=2, projid32bit=1
=                               crc=1       finobt=1, sparse=1, rmapbt=0
=                               reflink=1    bigtime=1 inobtcount=1 nrext64=0
data      =                    bsize=4096    blocks=5242880, imaxpct=25
naming    =                    suinit=0      swidth=0 blks
log        =                    bsize=4096    ssuinit=0, rtype=1
=                    blocks=16384, version=2
=                    sectsz=512   suinit=0 blks, lazy-count=1
realtime  =none                extsz=4096   blocks=0, rtextents=0
(root@192 lzc) # mkdir -p /data/brick1
(root@192 lzc) # echo '/dev/sdb /data/brick1 xfs defaults 1 2' >> /etc/fstab
(root@192 lzc) # mount -a && mount
mount: (hint) your fstab has been modified, but systemd still uses
the old version; use 'systemctl daemon-reload' to reload.
proc on /proc type proc (rw,nosuid,nodev,noexec,relatime)
sysfs on /sys type sysfs (rw,nosuid,nodev,noexec,relatime,seclabel)
devpts on /dev type devpts (rw,nosuid,seclabel,size=4096k,nr_inodes=244625,mode=755,inode64)
securityfs on /sys/kernel/security type securityfs (rw,nosuid,nodev,noexec,relatime)
tmpfs on /dev/shm type tmpfs (rw,nosuid,nodev,seclabel,inode64)
devpts on /dev/pts type devpts (rw,nosuid,noexec,relatime,seclabel,gid=5,mode=620,ptmxmode=000)
tmpfs on /run type tmpfs (rw,nosuid,nodev,seclabel,size=409612k,nr_inodes=819200,mode=755,inode64)
cgroup2 on /sys/fs/cgroup type cgroup2 (rw,nosuid,nodev,noexec,relatime,seclabel,nodelegat,memory_recursiveprot)
pstore on /sys/fs/pstore type pstore (rw,nosuid,nodev,noexec,relatime,seclabel)
bpf on /sys/fs/bpf type bpf (rw,nosuid,nodev,noexec,relatime,mode=700)
ramfs on /run/credentials/systemd-ocm-console-setup.service type ramfs (ro,nosuid,nodev,noexec,relatime,seclabel,mode=700)
configfs on /sys/kernel/config type configfs (rw,nosuid,nodev,noexec,relatime)
/dev/mapper/fedora_192-root on / type xfs (rw,relatime,seclabel,attr2,inode64,logbufs=8,logbsize=32k,noquota)
selinuxfs on /sys/fs/selinux type selinuxfs (rw,nosuid,noexec,relatime)
systemd-1 on /proc/sys/fs/binfmt_misc type autofs (rw,relatime,fd=33,pgrp=1,timeout=0,minproto=5,maxproto=5,direct,pipe_ino=16381)
hugetlbfs on /dev/hugepages type hugetlbfs (rw,relatime,seclabel,pagesize=2M)
mqueue on /dev/mqueue type mqueue (rw,nosuid,nodev,noexec,relatime,seclabel)
debugfs on /sys/kernel/debug type debugfs (rw,nosuid,nodev,noexec,relatime,seclabel)
tracefs on /sys/kernel/tracing type tracefs (rw,nosuid,nodev,noexec,relatime,seclabel)
fusectl on /sys/fs/fuse/connections type fusectl (rw,nosuid,nodev,noexec,relatime)
ramfs on /run/credentials/systemd-sysctl.service type ramfs (ro,nosuid,nodev,noexec,relatime,seclabel,mode=700)
ramfs on /run/credentials/systemd-tmpfiles-setup.service type ramfs (ro,nosuid,nodev,noexec,relatime,seclabel,mode=700)
ramfs on /run/credentials/systemd-tmpfiles-setup-dev.service type ramfs (ro,nosuid,nodev,noexec,relatime,seclabel,mode=700)
ramfs on /tmp type tmpfs (rw,nosuid,nodev,seclabel,size=1000020k,nr_inodes=1040576,inode64)
/dev/sda2 on /boot type xfs (rw,relatime,seclabel,attr2,inode64,logbufs=8,logbsize=32k,noquota)
ramfs on /run/credentials/systemd-tmpfiles-setup.service type ramfs (ro,nosuid,nodev,noexec,relatime,seclabel,mode=700)
ramfs on /run/credentials/systemd-resolved.service type ramfs (ro,nosuid,nodev,noexec,relatime,seclabel,mode=700)
sunrpc on /var/lib/nfs/rpc_pipefs type rpc_pipefs (rw,relatime)
tmpfs on /var/user/1000 type tmpfs (rw,nosuid,nodev,relatime,seclabel,size=200004k,nr_inodes=50001,mode=700,uid=1000,gid=1000,inode64)
/dev/sdb on /data/brick1 type xfs (rw,relatime,seclabel,attr2,inode64,logbufs=8,logbsize=32k,noquota)
```

2.5 安装 GlusterFS

在每个虚拟机节点上执行以下命令，以安装 GlusterFS：

```
yum install glusterfs-server
```

启动 GlusterFS 管理守护进程：

```
service glusterd start
service glusterd status
```

```
[root@192 lzc]# service glusterd start
Redirecting to /bin/systemctl start glusterd.service
[root@192 lzc]# service glusterd status
Redirecting to /bin/systemctl status glusterd.service
● glusterd.service - GlusterFS, a clustered file-system server
   Loaded: loaded (/usr/lib/systemd/system/glusterd.service; disabled; preset: disabled)
   Drop-In: /usr/lib/systemd/system/service.d
            └─10-timeout-abort.conf
   Active: active (running) since Sun 2024-04-21 22:22:09 CST; 9s ago
     Docs: man:glusterd(8)
   Process: 1598 ExecStart=/usr/sbin/glusterd -p /var/run/glusterd.pid --log-level $LOG_LEVEL $GLUSTERD_OPTIONS (code=exited, status=0/SUCCESS)
   Main PID: 1599 (glusterd)
     Tasks: 24 (limit: 2293)
    Memory: 15.6M
       CPU: 21ms
   CGroup: /system.slice/glusterd.service
           └─1599 /usr/sbin/glusterd -p /var/run/glusterd.pid --log-level INFO

Apr 21 22:22:09 localhost.localdomain systemd[1]: Starting glusterd.service - GlusterFS, a clustered file-system server...
Apr 21 22:22:09 localhost.localdomain systemd[1]: Started glusterd.service - GlusterFS, a clustered file-system server.
```

GlusterFS 管理守护进程启动成功！

关闭 GlusterFS 管理守护进程：

```
service glusterd stop
```

```
[root@192 lzc]# service glusterd stop
Redirecting to /bin/systemctl stop glusterd.service
[root@192 lzc]# service glusterd status
Redirecting to /bin/systemctl status glusterd.service
■ glusterd.service - GlusterFS, a clustered file-system server
   Loaded: loaded (/usr/lib/systemd/system/glusterd.service; disabled; preset: disabled)
   Drop-In: /usr/lib/systemd/system/service.d
            └─10-timeout-abort.conf
   Active: inactive (dead)
     Docs: man:glusterd(8)
```

GlusterFS 管理守护进程关闭成功！

2.6 配置可信池

首先，使用“systemctl stop firewalld.service”命令，关闭每个虚拟机的防火墙。

然后，使用命令“systemctl status firewalld”查看防火墙状态：

```
[lzc@localhost ~]$ systemctl stop firewalld.service
==== AUTHENTICATING FOR org.freedesktop.systemd1.manage-units ====
Authentication is required to stop 'firewalld.service'.
Authenticating as: lizhicheng (lzc)
Password:
==== AUTHENTICATION COMPLETE ====
[lzc@localhost ~]$ systemctl status firewalld
■ firewalld.service - firewalld - dynamic firewall daemon
   Loaded: loaded (/usr/lib/systemd/system/firewalld.service; enabled; preset: enabled)
   Drop-In: /usr/lib/systemd/system/service.d
            └─10-timeout-abort.conf
   Active: inactive (dead) since Sun 2024-04-21 23:25:07 CST; 1min 7s ago
 Duration: 2min 2.685s
     Docs: man:firewalld(1)
   Process: 901 ExecStart=/usr/sbin/firewalld --nofork --nopid $FIREWALLD_ARGS (code=exited, status=0/SUCCESS)
   Main PID: 901 (code=exited, status=0/SUCCESS)
      CPU: 806ms

Apr 21 23:23:02 localhost systemd[1]: Starting firewalld.service - firewalld - dynamic firewall daemon...
Apr 21 23:23:05 localhost systemd[1]: Started firewalld.service - firewalld - dynamic firewall daemon.
Apr 21 23:25:07 localhost.localdomain systemd[1]: Stopping firewalld.service - firewalld - dynamic firewall daemon...
Apr 21 23:25:07 localhost.localdomain systemd[1]: firewalld.service: Deactivated successfully.
Apr 21 23:25:07 localhost.localdomain systemd[1]: Stopped firewalld.service - firewalld - dynamic firewall daemon.
```

从终端显示可知，防火墙已被关闭。

在其中一个虚拟机节点上，使用如下命令，探测其他所有虚拟机节点：

```
gluster peer probe 192.168.61.131
gluster peer probe 192.168.61.132
```

在此实验中，我一共创建了三个虚拟机：Fedora_1(192.168.61.130)，Fedora_2(192.168.61.131)，Fedora_3(192.168.61.132)，此处，是以 Fedora_1 去探测 Fedora_2 和 Fedora_3，需要注意的是，这里需要提前开启 root 权限！


```
[root@localhost lzcl]# gluster peer probe 192.168.61.131
peer probe: success
[root@localhost lzcl]# gluster peer probe 192.168.61.132
peer probe: success
```

根据终端显示信息可知，探测成功！

2.7 建立 GlusterFS 卷

在虚拟机 Fedora_2 和 Fedora_3 上创建文件夹：

```
mkdir /data/brick1/gv0
```

在虚拟机 Fedora_1 上运行：

```
gluster volume create gv0 replica 2 192.168.61.131:/data/brick1/gv0
192.168.61.132:/data/brick1/gv0

gluster volume start gv0
```

```
[root@localhost lzcl]# gluster volume create gv0 replica 2 192.168.61.131:/data/brick1/gv0 192.168.61.132:/data/brick1/gv0
Replica 2 volumes are prone to split-brain. Use Arbiter or Replica 3 to avoid this. See: http://docs.gluster.org/en/latest/Administrator-Guide/Split-brain-and-ways-to-deal-with-it/.
Do you still want to continue?
(y/n) y
volume create: gv0: success: please start the volume to access data
[root@localhost lzcl]# gluster volume start gv0
volume start: gv0: success
```

执行这个命令后，GlusterFS 会创建一个名为 gv0 的新卷，该卷跨越两个服务器节点，每个节点上都有一个 brick，数据会在这两个 brick 之间进行复制，确保了副本数为 2。这意味着任何写入 gv0 卷的数据都会同时保存在两个节点上的 gv0 目录中。

使用副本卷可以提高数据的可靠性，因为它允许从任一副本上读取数据，如果其中一个 brick 发生故障，另一个可以作为备份继续提供服务。在实际部署中，这两个节点应该位于不同的物理机器上。

确认卷的信息：

```
gluster volume info
```

```
[root@localhost lzcl]# gluster volume info

Volume Name: gv0
Type: Replicate
Volume ID: 3e634ea8-801b-475e-8f5b-c78b1ba908b6
Status: Started
Snapshot Count: 0
Number of Bricks: 1 x 2 = 2
Transport-type: tcp
Bricks:
Brick1: 192.168.61.131:/data/brick1/gv0
Brick2: 192.168.61.132:/data/brick1/gv0
Options Reconfigured:
cluster.granular-entry-heal: on
storage.fips-mode-rchecksum: on
transport.address-family: inet
nfs.disable: on
performance.client-io-threads: off
```

确认信息与预期一致！


```
service glusterd stop
service rpcbind start      #开启 rpcbind 服务
service glusterd start    #必须重启 glusterd 服务
```

我们将使用其中一个服务节点，挂载该卷。通常，我们从外部的机器中挂载该卷，成为客户端。由于该方法需要安装额外的包到客户端机器上，因此我们将使用其中一个服务节点作为“客户端”，进行简单的测试。

```
mount -t glusterfs 192.168.61.131:gv0 /mnt
```

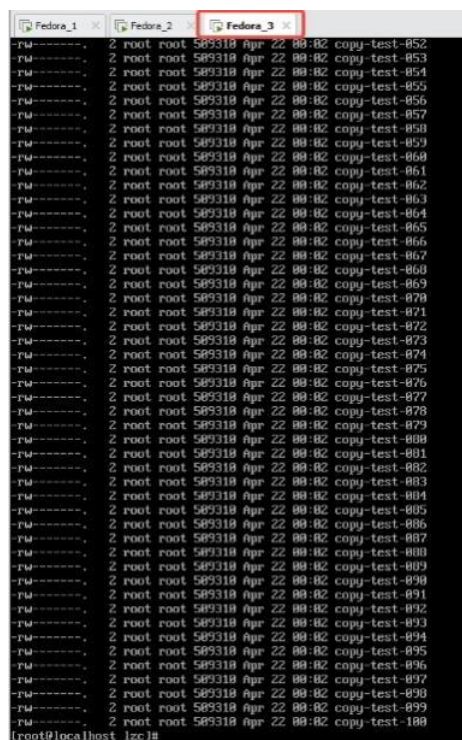
```
for i in `seq -w 1 100`; do cp -rp /var/log/messages /mnt/copy-test-$i; done
```

```
ls -lA /mnt | wc -l
```

```
ls -lA /data/brick1/gv0
```

[illegible]

虚拟机 Fedora_3:



```
root@localhost ~# ls -l /root/.ssh/
-rw-r--r-- 1 root root 589318 Apr 22 00:02 copy-test-052
-rw-r--r-- 1 root root 589318 Apr 22 00:02 copy-test-053
-rw-r--r-- 1 root root 589318 Apr 22 00:02 copy-test-054
-rw-r--r-- 1 root root 589318 Apr 22 00:02 copy-test-055
-rw-r--r-- 1 root root 589318 Apr 22 00:02 copy-test-056
-rw-r--r-- 1 root root 589318 Apr 22 00:02 copy-test-057
-rw-r--r-- 1 root root 589318 Apr 22 00:02 copy-test-058
-rw-r--r-- 1 root root 589318 Apr 22 00:02 copy-test-059
-rw-r--r-- 1 root root 589318 Apr 22 00:02 copy-test-060
-rw-r--r-- 1 root root 589318 Apr 22 00:02 copy-test-061
-rw-r--r-- 1 root root 589318 Apr 22 00:02 copy-test-062
-rw-r--r-- 1 root root 589318 Apr 22 00:02 copy-test-063
-rw-r--r-- 1 root root 589318 Apr 22 00:02 copy-test-064
-rw-r--r-- 1 root root 589318 Apr 22 00:02 copy-test-065
-rw-r--r-- 1 root root 589318 Apr 22 00:02 copy-test-066
-rw-r--r-- 1 root root 589318 Apr 22 00:02 copy-test-067
-rw-r--r-- 1 root root 589318 Apr 22 00:02 copy-test-068
-rw-r--r-- 1 root root 589318 Apr 22 00:02 copy-test-069
-rw-r--r-- 1 root root 589318 Apr 22 00:02 copy-test-070
-rw-r--r-- 1 root root 589318 Apr 22 00:02 copy-test-071
-rw-r--r-- 1 root root 589318 Apr 22 00:02 copy-test-072
-rw-r--r-- 1 root root 589318 Apr 22 00:02 copy-test-073
-rw-r--r-- 1 root root 589318 Apr 22 00:02 copy-test-074
-rw-r--r-- 1 root root 589318 Apr 22 00:02 copy-test-075
-rw-r--r-- 1 root root 589318 Apr 22 00:02 copy-test-076
-rw-r--r-- 1 root root 589318 Apr 22 00:02 copy-test-077
-rw-r--r-- 1 root root 589318 Apr 22 00:02 copy-test-078
-rw-r--r-- 1 root root 589318 Apr 22 00:02 copy-test-079
-rw-r--r-- 1 root root 589318 Apr 22 00:02 copy-test-080
-rw-r--r-- 1 root root 589318 Apr 22 00:02 copy-test-081
-rw-r--r-- 1 root root 589318 Apr 22 00:02 copy-test-082
-rw-r--r-- 1 root root 589318 Apr 22 00:02 copy-test-083
-rw-r--r-- 1 root root 589318 Apr 22 00:02 copy-test-084
-rw-r--r-- 1 root root 589318 Apr 22 00:02 copy-test-085
-rw-r--r-- 1 root root 589318 Apr 22 00:02 copy-test-086
-rw-r--r-- 1 root root 589318 Apr 22 00:02 copy-test-087
-rw-r--r-- 1 root root 589318 Apr 22 00:02 copy-test-088
-rw-r--r-- 1 root root 589318 Apr 22 00:02 copy-test-089
-rw-r--r-- 1 root root 589318 Apr 22 00:02 copy-test-090
-rw-r--r-- 1 root root 589318 Apr 22 00:02 copy-test-091
-rw-r--r-- 1 root root 589318 Apr 22 00:02 copy-test-092
-rw-r--r-- 1 root root 589318 Apr 22 00:02 copy-test-093
-rw-r--r-- 1 root root 589318 Apr 22 00:02 copy-test-094
-rw-r--r-- 1 root root 589318 Apr 22 00:02 copy-test-095
-rw-r--r-- 1 root root 589318 Apr 22 00:02 copy-test-096
-rw-r--r-- 1 root root 589318 Apr 22 00:02 copy-test-097
-rw-r--r-- 1 root root 589318 Apr 22 00:02 copy-test-098
-rw-r--r-- 1 root root 589318 Apr 22 00:02 copy-test-099
-rw-r--r-- 1 root root 589318 Apr 22 00:02 copy-test-100
```

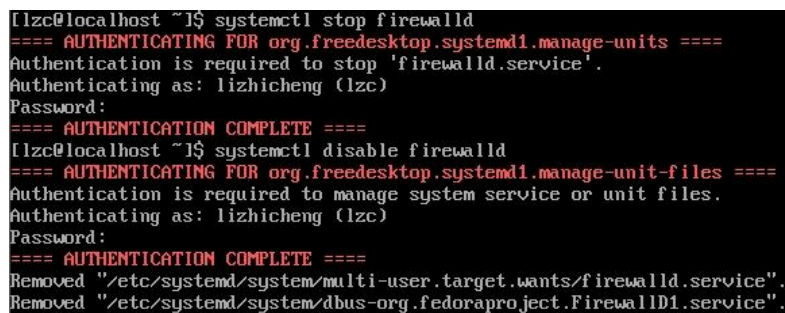
3 实验二：搭建 ownCloud

本次实验环境基于搭建 GlusterFS 的 Fedora 服务器，选择其中任意一台即可。这里，我选择虚拟机 Fedora_2。

3.1 准备工作

3.1.1 关闭防火墙

```
systemctl stop firewalld
systemctl disable firewalld
```



```
[lzc@localhost ~]$ systemctl stop firewalld
==== AUTHENTICATING FOR org.freedesktop.systemd1.manage-units ====
Authentication is required to stop 'firewalld.service'.
Authenticating as: lizhicheng (lzc)
Password:
==== AUTHENTICATION COMPLETE ====
[lzc@localhost ~]$ systemctl disable firewalld
==== AUTHENTICATING FOR org.freedesktop.systemd1.manage-unit-files ====
Authentication is required to manage system service or unit files.
Authenticating as: lizhicheng (lzc)
Password:
==== AUTHENTICATION COMPLETE ====
Removed "/etc/systemd/system/multi-user.target.wants/firewalld.service".
Removed "/etc/systemd/system/dbus-org.fedoraproject.FirewallD1.service".
```

防火墙已关闭！

3.1.2 关闭 selinux

修改 /etc/sysconfig/selinux，将 **SELINUX=disable**。

```
[root@localhost lzc]# cat /etc/sysconfig/selinux
# This file controls the state of SELinux on the system.
# SELINUX= can take one of these three values:
#   enforcing - SELinux security policy is enforced.
#   permissive - SELinux prints warnings instead of enforcing.
#   disabled - No SELinux policy is loaded.
# See also:
# https://docs.fedoraproject.org/en-US/quick-docs/getting-started-with-selinux/#getting-started-with-selinux-selinux-states-and-modes
# NOTE: In earlier Fedora kernel builds, SELINUX=disabled would also
# fully disable SELinux during boot. If you need a system with SELinux
# fully disabled instead of SELinux running with no policy loaded, you
# need to pass selinux=0 to the kernel command line. You can use grubby
# to persistently set the bootloader to boot with selinux=0:
#
#   grubby --update-kernel ALL --args selinux=0
#
# To revert back to SELinux enabled:
#
#   grubby --update-kernel ALL --remove-args selinux
#
SELINUX=disabled
# SELINUXTYPE= can take one of these three values:
#   targeted - Targeted processes are protected.
#   minimum - Modification of targeted policy. Only selected processes are protected.
#   mls - Multi Level Security protection.
SELINUXTYPE=targeted
```

cat 检查一下，发现已经成功修改！

3.1.3 更新 yum 源

```
# 备份当前的 YUM 源配置文件：
cp /etc/yum.repos.d/fedora.repo /etc/yum.repos.d/fedora.repo.bak
cp /etc/yum.repos.d/fedora-updates.repo /etc/yum.repos.d/fedora-updates.repo.bak

dnf upgrade --refresh # 更新 yum 源配置文件

reboot # 重启系统使配置文件生效
```

3.2 安装 ownCloud

3.2.1 安装 samba、httpd

```
yum install -y samba
yum install -y httpd
```

3.2.2 安装 php7.4

安装 remi 源：

```
yum -y install http://mirrors.tuna.tsinghua.edu.cn/remi/fedora/remi-release-38.rpm
```

```
[root@localhost lzc]# yum -y install http://mirrors.tuna.tsinghua.edu.cn/remi/fedora/remi-release-38.rpm
Failed to set locale, defaulting to C.UTF-8
Last metadata expiration check: 0:19:47 ago on Mon Apr 22 12:38:48 2024.
remi-release-38.rpm
Dependencies resolved.
=====
Package Architecture Version Repository Size
Installing:
remi-release noarch 38-5.fc38.remi @commandline 31 k
Transaction Summary
-----
Install 1 Package
Total size: 31 k
Installed size: 29 k
Downloading Packages:
Running transaction check
Transaction check succeeded.
Running transaction test
Transaction test succeeded.
Running transaction
  Preparing      :
  Installing     : remi-release-38-5.fc38.remi.noarch 1/1
  Verifying      : remi-release-38-5.fc38.remi.noarch 1/1
Installed:
remi-release-38-5.fc38.remi.noarch
Complete!
```

修改 /etc/yum.repos.d/remi.repo:

```
[remi]
name=Remi's RPM repository - Fedora $releasever - $basearch
baseurl=http://mirrors.tuna.tsinghua.edu.cn/remi/fedora/$releasever/remi/$basearch/
enabled=1
gpgcheck=0
```

cat 检查一下, 发现已经成功修改!

```
[root@localhost lzcl]# cat /etc/yum.repos.d/remi.repo
# Repository: https://rpms.remirepo.net/
# Blog:      https://blog.remirepo.net/
# Forum:     https://forum.remirepo.net/

[remi]
name=Remi's RPM repository - Fedora $releasever - $basearch
#baseurl=https://rpms.remirepo.net/fedora/$releasever/remi/$basearch/
baseurl=http://mirrors.tuna.tsinghua.edu.cn/remi/fedora/$releasever/remi/$basearch
mirrorlist=http://cdn.remirepo.net/fedora/$releasever/remi/$basearch/mirror
enabled=1
gpgcheck=0
# can be enabled if not being a proxy because of possible cache issue
repo_gpgcheck=0
fastestmirror=1
gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY-remi-$releasever
```

安装 php7.4:

```
yum -y install --enablerepo=remi php74-php-fpm php74 php74-php php74-php-opcache
php74-php-xml php74-php-mcrypt php74-php-gd php74-php-devel php74-php-mysql php74-php-intl
php74-php-mbstring php74-php-zip
```

查看 php 版本:

```
php74 -v
```

```
[root@localhost lzcl]# php74 -v
PHP 7.4.33 (cli) (built: Apr 10 2024 09:17:01) ( NTS )
Copyright (c) The PHP Group
Zend Engine v3.4.0, Copyright (c) Zend Technologies
with Zend OPcache v7.4.33, Copyright (c), by Zend Technologies
```

3.2.3 安装 mariadb

```
yum install -y mariadb mariadb-server
```

3.2.4 设置 samba 开机启动

```
systemctl start smb.service
systemctl enable smb.service
```

```
[root@localhost lzcl]# systemctl start smb.service
[root@localhost lzcl]# systemctl enable smb.service
Created symlink /etc/systemd/system/multi-user.target.wants/smb.service → /usr/lib/systemd/system/smb.service.
```

3.2.5 设置 httpd 开机启动

```
systemctl start httpd.service  
systemctl enable httpd.service
```

```
[root@localhost ~]# systemctl start httpd.service  
[root@localhost ~]# systemctl enable httpd.service  
Created symlink /etc/systemd/system/multi-user.target.wants/httpd.service → /usr/lib/systemd/system/httpd.service.
```

3.2.6 设置 mariadb 开机启动

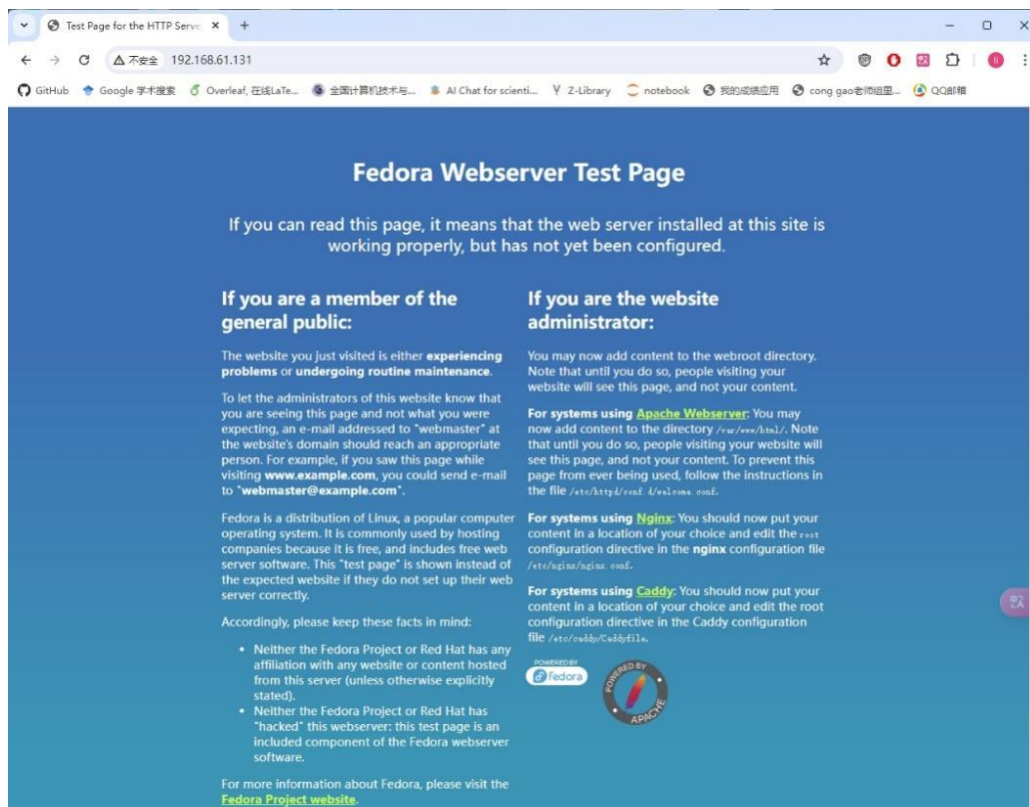
```
systemctl start mariadb.service  
systemctl enable mariadb.service
```

```
[root@localhost ~]# systemctl start mariadb.service  
[root@localhost ~]# systemctl enable mariadb.service  
Created symlink /etc/systemd/system/mysqld.service → /usr/lib/systemd/system/mariadb.service.  
Created symlink /etc/systemd/system/mysqld.service → /usr/lib/systemd/system/mariadb.service.  
Created symlink /etc/systemd/system/multi-user.target.wants/mariadb.service → /usr/lib/systemd/system/mariadb.service.
```

3.2.7 修改 /var/www/html 权限

```
chown apache:apache /var/www/html/
```

在虚拟机外部打开浏览器，网址栏处输入虚拟机 Fedora_2 的 IP 地址(192.168.61.131)，可以看到 apache 测试页：



3.2.8 安装 ownCloud

```
cd /var/www/html
```


下载 ownCloud:

```
wget https://download.owncloud.com/server/stable/owncloud-10.10.0.tar.bz2
```

解压:

```
tar -xjvf owncloud-10.10.0.tar.bz2
```

拷贝 index.php 到 html 下:

```
cp /var/www/html/owncloud/index.php /var/www/html/  
ls
```

```
[root@localhost html]# ls  
index.php  owncloud  owncloud-10.10.0.tar.bz2
```

修改权限:

```
chmod 777 /var/www/html/owncloud
```

3.2.9 重启 httpd

```
systemctl restart httpd.service  
ps -aux | grep httpd
```

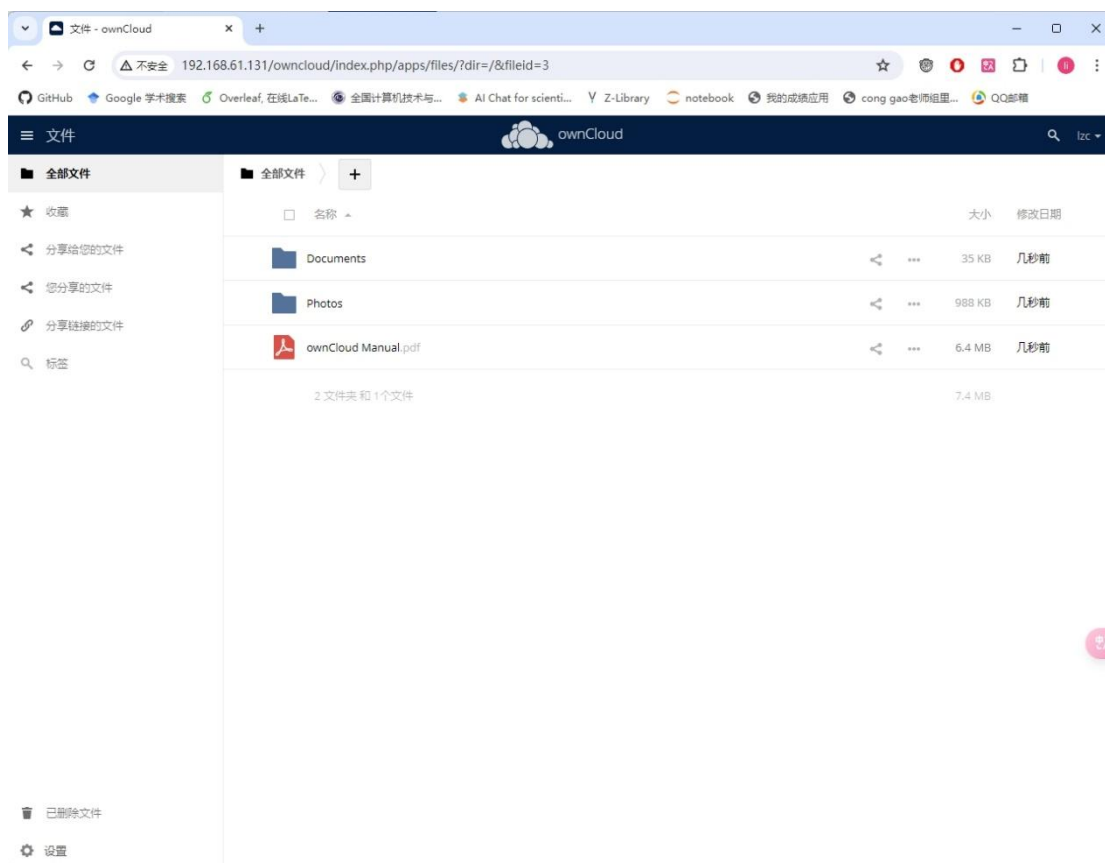
```
[root@localhost html]# ps -aux | grep httpd  
root      3039  0.1  0.5 18140 11012 ?        Ss   15:27   0:00 /usr/sbin/httpd -DFOREGROUND  
apache    3041  0.0  0.3 18212  6496 ?        S    15:27   0:00 /usr/sbin/httpd -DFOREGROUND  
apache    3042  0.0  0.4 2222820 8492 ?        Sl   15:27   0:00 /usr/sbin/httpd -DFOREGROUND  
apache    3043  0.0  0.4 2419492 8752 ?        Sl   15:27   0:00 /usr/sbin/httpd -DFOREGROUND  
apache    3044  0.0  0.4 2222820 8248 ?        Sl   15:27   0:00 /usr/sbin/httpd -DFOREGROUND  
root      3226  0.0  0.0   3276  1692 tty1    S+   15:27   0:00 grep --color=auto httpd
```

3.3 注册测试

在虚拟机外部打开浏览器，输入“192.168.61.131/owncloud”:



创建管理员账号，并登录。



登录成功！

4 思考并拓展实践

要为 ownCloud 用户存储的文档提供高可靠和高可用的服务，可以根据 GlusterFS 的七种卷类型的特性进行选择和配置。这些卷类型包括：分布式卷、复制卷、条带卷、分布式复制卷、分布式条带卷、条带复制卷和分布式条带复制卷。下面将详细介绍如何结合这些卷类型为 ownCloud 提供优化的存储解决方案：

分布式卷：适用于大量非关键数据存储，可以提供良好的扩展性，但不提供数据冗余。

复制卷：通过数据复制提供高可靠性。每个数据块都有一个或多个副本，如果一个服务器失败，数据可以从其他服务器上的副本中恢复。这对于需要高数据可靠性的 ownCloud 环境非常合适。

条带卷：将数据分割成多个块，分布在不同的服务器上，适合大型文件的存储，可以提高读写速度，但不提供数据冗余。

分布式复制卷：结合了分布式卷和复制卷的特点，提供了数据扩展性和冗余性。这对于 ownCloud 用户是理想的选择，因为它既提供了存储容量的可扩展性，又确保了数据的可靠性。

分布式条带卷：增加了扩展性和读写性能，适用于存储大型文件，但同样不提供数据冗余。

条带复制卷：将条带卷和复制卷的特点结合起来，提供了高性能和数据冗余。适用于需要高性能和高可靠性的大型文件存储。

分布式条带复制卷：是条带复制卷和分布式卷的结合，提供最优的性能和扩展性，同时

也提供数据冗余，适合于非常大的数据集，需要高性能和高可靠性的环境。

具体实施建议：

评估数据的重要性：对于关键数据，应使用复制卷或分布式复制卷以确保高可靠性。

考虑数据的大小和访问模式：对于大型文件，可以考虑使用条带卷或分布式条带卷来提高性能。

容错和灾难恢复：应配置足够的副本数（通常为 2 或更多），并在不同的物理位置部署服务器以应对硬件故障和自然灾害。

性能与成本的平衡：复制和条带技术可以提高性能和可靠性，但也会增加存储需求和成本。合理规划存储架构可以帮助平衡这些因素。

通过这种方式，可以为 ownCloud 用户提供一个既高效又可靠的存储解决方案，确保数据的安全性和可用性。