rsync数据同步服务

作者：牟建波（1353429820@qq.com）

时间：2025-05-29

描述：日常自学笔记

1.数据同步服务介绍

在CentOS Stream 9中，数据同步服务Rsync是一个开源的多功能的用于在本地和远程系统之间高效同步和备份文件的工具，通过只传输变化的数据来节省带宽和时间

- 支持增量同步：只传输发生变化的数据部分
- 支持SSH加密传输
- 可保持文件权限、时间戳、连接和等元数据

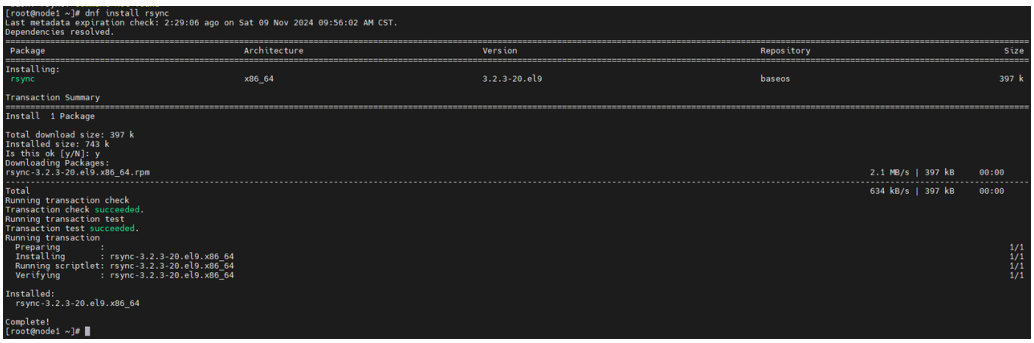
代码块

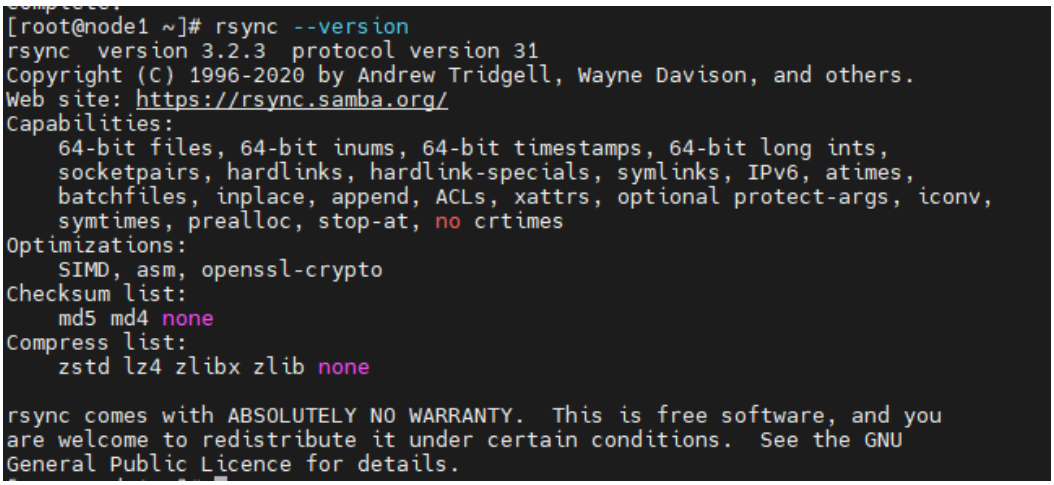
```
Rsync的基本原理是通过使用“差异算法”来仅同步源文件和目标文件之间的差异部分，而不是重新传输整个文件

这一过程的核心是：
    1.分块比较：Rsync将文件分为多个小块（通常是固定大小的块），然后对比源文件和目标文件中每个块的内容
    2.校验和计算：对于每个块，Rsync计算出一个校验和（checksum），然后将源文件和目标文件的校验和进行比较，判断哪些块发生了变化
    3.传输差异部分：如果某块在源文件和目标文件中不同，则只传输这一部分的差异数据，而不是整个文件
    4.优化：Rsync还支持压缩（通过-z选项）和增量传输，使得同步过程更加高效，尤其在带宽有限的情况下

通过这种机制，Rsync大大减少了传输的数据量，从而提高了文件同步的效率，特别是在大文件和大规模目录结构中
```

2.安装数据同步服务

代码块

```
# 数据同步需要双方按照rsync，同事配置SSH密钥，否则需要输入密码
dnf install -y rsync
rsync --version
```





3.rsync基本语法

代码块

```
rsync [选项] 源路径... 目标路径

常见选项：
    -v：详细模式输出，显示传输过程中的文件信息（常用）
    -a：归档模式，表示递归传输文件，并保存所有文件属性（常用）
    -z：对备份的文件在传输时进行压缩处理
    -P：显示进度条
    --delete：删除目标目录中源目录中没有的文件（可选，用于实现双向同步）
    -e：用于指定用于传输文件的远程shell程序，默认ssh，也可以选择其他

# 常用举例
```

```
rsync -av /src/directory/ /dest/directory/                          # 本地同步
rsync -avz /src/directory/ root@192.168.88.102:/dest/directory/ # 远程同步
rsync -avz user@192.168.88.102:/src/directory/ /dest/directory/ # 拉取远程数据到本地
```

本地同步案例

代码块

```
# 本地同步案例（node1）
环境：
    1.node1节点/usr/local目录下创建nginx-1.8.0目录，并在其下创建一个html目录
    2.html目录中创建index.html、favicon.icon、logo.png、assets目录
    3.assets目录中创建index.js、index.css
    4.给各个文件写入一些内容

# 环境准备
cd /usr/local
mkdir -p nginx-1.8.0/html
cd nginx-1.8.0/html
touch index.html favicon.ico logo.png
mkdir assets
cd assets
touch index.js index.css
cd /usr/local/nginx-1.8.0/html
echo "index.html" >> index.html
echo "favicon.ico" >> favicon.ico
echo "logo.png" >> logo.png
cd /usr/local/nginx-1.8.0/html/assets
echo "index.js" >> index.js
echo "index.css" >> index.css

# 校验文件环境
tree /usr/local/nginx-1.8.0
nginx-1.8.0/
└── html
    ├── assets
    │   ├── index.css
    │   └── index.js
    ├── favicon.ico
    ├── index.html
    └── logo.png

# 需求：对html目录进行同步操作，将其同步到/usr/local/nginx-1.8.0/backup
cd /usr/local/nginx-1.8.0/backup    # 发现不存在
```

```
mkdir -p /usr/local/nginx-1.8.0/backup

# 同步操作
rsync -av --delete /usr/local/nginx-1.8.0/html /usr/local/nginx-1.8.0/backup/

# 校验同步情况
tree /usr/local/nginx-1.8.0/backup
backup/
└── html
    ├── assets
    │   ├── index.css
    │   └── index.js
    ├── favicon.ico
    ├── index.html
    └── logo.png

# 常见问题：backup没有index目录，因为nginx-1.8.0/html写成了nginx-1.8.0/html/，没有包含htnl本身
```

远程同步案例

代码块

```
# 远程同步案例
要求：node1中的/usr/local/nginx-1.8.0/html目录同步到node2中~/backup/

# 创建环境
[node1]
ssh root@192.168.88.102 "midir -p ~/bakcup"

# 同步操作
rsync -avz /usr/local/nginx-1.8.0/html root@192.168.88.102:~/bakcup

# 校验同步结果
ssh root@192.168.88.102 "tree ~/backup"
/root/backup
└── html
    ├── assets
    │   ├── index.css
    │   └── index.js
    ├── favicon.ico
    ├── index.html
    └── logo.png
```

4.常用操作：定时与定量备份

需求说明：在 `centos stream 9` 的 `/var/log` 目录下， 存储了大量的关于系统的日志：如messages文件存储大量的系统日志信息，secure日志文件存储了大量的与系统安全的日志。请对此目录中的 `messages` 、 `secure` 日志文件 `进行定时的备份` 到 `node2` 的 `/export/data/node1_backup_system_log` 目录中， 要求 `每日凌晨2点进行备份一次`

代码块

```
[node1]
# 创建node2中/export/data/node1_backup_system_log目录
ssh root@192.168.88.102 "mkdir -p /export/data/node1_backup_system_log;tree /export"

# 同步命令
rsync -avz /var/log/message /var/log/secure root@192.168.88.102:/export/data/node1_bakcup_system_log

# 定时操作：crontab
# 格式说明
crontab [选项]
    -l：查看当前有哪些定时任务
    -e：打开定时任务编辑窗口，设定定时任务

# 定时任务文件格式：分(0~59) 时(0~23) 日(1~31) 月(1~12) 周(1~7) 执行的命令
# 相关符号：*(任意)、/(每隔多少时间)、-(一段连续的周期)、,(表示某个时间内多个值)

# 编写定时任务：要求每日凌晨2点进行一次备份
crontab -e
0 2 * * * /usr/bin/rsync -avz /var/log/messages /var/log/secure root@192.168.88.102:/export/data/node1_backup_system_log
```

5.rsync结合inotify实现实时同步

需求说明：目前在企业服务 `node1服务器` 中， `/export/data/logs` 目录是公司各个系统日志存储目录，公司希望能够实时监控备份这些日志数据到 `node2` 的 `/export/data/node1_exe_backup_log` 目录下

5.1 inotify介绍

`inotify`：是Linux内核的一个子系统，用于监控文件系统事件。它允许应用程序实时监控文件或目录的变化，如创建、删除、修改等。inotify通过内核通知机制，避免了轮询文件系统的高开销，提升了效率

- 在 Shell 中，可以使用 `inotify-tools` 工具包来监控文件系统事件。`inotify-tools` 提供了两个主要的命令行工具：`inotifywait` 和 `inotifywatch`，其中 `inotifywait` 是最常用的工具，用于实时监控文件或目录的变化

代码块

```
# 安装inotify
dnf install -y inotify-tools

# inotify-tools工具可能没有直接提供在默认仓库中，安装epel仓库
dnf install -y epel-release
```

inotify使用方式

代码块

```
格式：inotifywait [选项] 文件或目录
常用选项
    -m：持续监控（默认只监控一次后退出）
    -r：递归监控目录及其子目录
    -e：指定要监控的事件类型（如 create、delete、modify 等）
    -q：静默模式，减少输出信息
    --format：自定义输出格式

常用事件类型
    create：文件或目录被创建。
    delete：文件或目录被删除。
    modify：文件内容被修改。
    attrib：文件属性（如权限）被修改。
    move：文件或目录被移动。
```

代码块

```
# 监控目录中的文件创建和删除事件
inotifywait -m -e create,delete /path/dir

# 递归监控目录的文件修改事件
inotifywait -m -r -e modify /path/dir
```

结合Shell脚本处理事件

代码块

```
#!/bin/bash
# 第一行为脚本文件的固定开头， 必须要携带的
# 要监控的目录
DIR="/path/to/dir"

inotifywait -m -e create,delete,modify,attrib "$DIR" | while read -r path action file;
do
    if [[ "$action" == *"CREATE"* ]]; then
        echo "File created: $file"
        # 执行创建文件后的操作
    elif [[ "$action" == *"DELETE"* ]]; then
        echo "File deleted: $file"
        # 执行删除文件后的操作
    elif [[ "$action" == *"MODIFY"* ]]; then
        echo "File MODIFY: $file"
        # 执行文件内容被修改后的操作
    elif [[ "$action" == *"ATTRIB"* ]]; then
        echo "File ATTRIB: $file"
        # 执行文件被修改属性后的操作
    fi
done
```

5.2 配置inotify客户端实现实时同步

代码块

```
# 1.编写inotify监控脚本
vim /root/real_time_sync.sh

添加以下内容：
#!/bin/bash
```

```
# 定义源目录和目标目录（目标目录是远程服务器的目录）
SOURCE_DIR="/export/data/logs"
TARGET_DIR="root@192.168.88.102:/export/data/node1_exe_backup_log"

# 日志文件
LOG_FILE="/var/log/rsync_realtime_sync.log"

# 使用inotifywait监听${WATCH_DIR}目录中的文件变化
inotifywait -m -r -e modify,create,delete,move "${SOURCE_DIR}" | while read path action file;
do
    echo "Detected ${action} on ${file} in ${path}. Starting rsync..." >> "$LOG_FILE"
    # 执行rsync同步操作
    rsync -avz --delete -e ssh "${SOURCE_DIR}/" "${TARGET_DIR}"  >> "$LOG_FILE" 2>&1
    echo "Sync completed for ${file} in ${path}." >> "$LOG_FILE"
done

# 2.设置脚本可执行权限
chmod +x /root/real_time_sync.sh

# 3.测试是否可以实时同步
/root/real_time_sync.sh
# 往node1中的/export/data/logs路径放入一个文件，看node2是否同步
touch inotify_test    # 测试成功

# 4.后台挂载同步脚本（服务器重启后，需要手动启动）
nohup /root/real_time_sync &

# 5.使用systemd创建同步服务
vim /etc/systemd/system/rsync_inotify.service
内容如下：
[Unit]
Description=Real-time Rsync Sync with Inotify
After=network.target

[Service]
ExecStart=/root/real_time_sync.sh
Restart=always
User=root
Group=root

[Install]
WantedBy=multi-user.target

保持退出后：
```

```
systemctl daemon-reload
systemctl start rsync_inotify
systemctl enable rsync_inotify
```

`inotifywait -m -r -e create,modify,delete,move`：这个命令用来监听源目录的创建、修改、删除和移动事件。`-m` 参数表示持续监听，`-r` 表示递归地监听子目录。

`rsync -avz --delete -e ssh`：当监测到文件变化时，使用 `rsync` 同步源目录到远程目标目录。`-e ssh` 表示使用 SSH 连接远程服务器进行同步，`--delete` 参数保证如果源目录删除了文件，目标目录也会同步删除这些文件。

日志记录：每次文件变化和同步都会记录到 `/var/log/rsync_realtime_sync.log` 中，方便查看。

5.3 基于rsync服务方式实现同步

node2设置rsync为服务项：设置Rsync为服务型，指的是让Rsync已守护进程（daemo）的模式运行，在后台持续运行，等待来自客户端的连接和同步请求

- 优点：高效、简化配置流程
- 缺点：安全性不高、需要开放端口、配置文件复杂

代码块

```
# 1.配置rsyncd.conf
vim /etc/rsyncd.conf

# 全局设置
uid = rsync
gid = rsync
use chroot = no
port = 873
max connections = 200
pid file = /var/run/rsyncd.pid
log file = /var/log/rsyncd.log
transfer logging = yes
ignore nonreadable = yes

[node1_logs]
    path = /export/data/node1_exe_backup_log
    comment = Backup Directory
    read only = no
    secrets file = /etc/rsyncd.passwd
    hosts allow = 192.168.88.0/24
    hosts deny = *
```

```
22      auth users = backup_user
```

- 各个参数详细说明

代码块

```
# 各个参数详细说明
# 全局设置
uid = rsync       # 运行rsync守护进程的用户
gid = rsync       # 运行rsync守护进程的组
use chroot = no   # 是否使用chroot环境，一般设置为no
port = 873 # 指定Rsync守护进程监听的端口号，默认是873。
max connections = 200  # 最大连接数
pid file = /var/run/rsyncd.pid  # 进程ID文件位置
log file = /var/log/rsyncd.log  # 日志文件位置
transfer logging = yes # 启用传输日志记录，记录每次文件传输的详细信息
ignore nonreadable = yes # 忽略无法读取的文件，不尝试同步它们

# 模块定义
[node1_logs]
    path = /export/data/node1_exe_backup_log    # 同步的目录
    comment = Backup Directory  # 模块描述
    read only = no # 设置为false/no允许写操作
    secrets file = /etc/rsyncd.passwd  # 密码文件位置
    hosts allow = 192.168.88.0/24  # 允许访问的IP地址段
    hosts deny = * # 拒绝所有其他主机的访问
    auth users = backup_user # 认证同步的用户必须是backup_user，如果不配置可以是任意用户

请注意，将uid和gid设置为root以及auth users设置为root可能带来安全风险，因为这意味着任何通过认证的用户都将能够以root权限访问和修改文件。在生产环境中，通常建议创建专门的用户和组来运行Rsync守护进程，并限制对敏感文件的访问。

注意:
    全局配置的用户， 是指的用于管理服务进程，并且执行相关的同步操作
    在模块中定义的用户， 用于基于这个模块来干活的用户是谁， 如果不是这个用户， 无法读取这个模块相关配置

此处采用的rsync,建议可以先创建这两个用户:
useradd rsync
useradd backup_user

并为其设置密码:  均为123456
```

代码块

```
# 2.创建密码文件：在配置文件中指定的secrets file需要包含用户和密码，格式为username:password
echo "backup_user:123456" > /etc/rsyncd.passwd

# 设置密码文件的权限为600，确保该文件只有 RSYNC 服务可以读取
chmod 600 /etc/rsyncd.passwd

# 3.启动rsync服务
rsync --daemon

# 4.测试访问
```

```
[node1]
rsync rsync://backup_user@192.168.88.102/node1_logs

[node2]：需要开放端口
firewall-cmd --zone=public --add-port=873/tcp --permanent
firewall-cmd --reload

# 5.查看并关闭rsync服务，后续设置开机启动
ps -ef | grep rsunc
kill -9 rsync的PID

# 6.设置开机启动
vim /etc/systemd/system/rsyncd.service
# 添加以下内容
[Unit]
Description=Rsync Daemon
After=network.target

[Service]
ExecStart=/usr/bin/rsync --daemon --config=/etc/rsyncd.conf --no-detach
ExecReload=/bin/kill -HUP $MAINPID # 通过发送SIGHUP信号，告诉rsync守护进程重新加载其配置文件，而无需停止并重新启动进程
Restart=always

[Install]
WantedBy=multi-user.target

# 7.重新加载system配置并启用rsync
systemctl daemon-reload
systemctl enable --now rsyncd
systemctl start rsyncd

# 8.验证服务状态
systemnctl status rsyncd
```

基于rsync服务进行数据同步

代码块

```
# 连接rsync服务的格式
用户名@主机地址::模块名称   或   rsync://用户名@主机地址/模块名称

说明：
    模块名：指的是配置到rsync.conf中【】的内容
```

```
    用户名：指的是模块中指定的用户名，如果没有指定，可以是主机存在的用户名即可（前提是该用户需要有权限）

例如：
    rsync://backup_user@192.168.88.102/node1_logs
    backup_user@192.168.88.102::node1_logs

# 基于手动同步方式完成数据同步
# 注意：在执行手动数据同步之前，需要先在/export/data/logs/创建了日志文件， 以保证在同步的时候，有数据
echo '12345678' > /export/data/logs/a.log
rsync -avz /export/data/logs backup_user@node2::node1_logs

# 报错改为node2的IP地址
rsync -avz /export/data/logs backup_user@192.168.88.102::node1_logs
```

代码块

```
# 可能出现的错误：node2 873：Name or service not known
# 原因：系统识别不到node2，并不知道node2对应的服务器是谁
# 解决方法：更改为IP地址 或 配置node2对应的IP映射关系

# 配置node2对应的映射关系
vim /etc/hosts
# 添加以下内容
192.168.88.101 node1
192.168.88.102 node2

# 可能出现的问题：写入node2的目录没有对应的权限，导致同步失败
[node2]
chown -R rsync:rsync /export/data/node1_exe_backup_log

# 优化：执行操作时，每次需要设置密码，如何解决?
可以通过rsync中配置：--password-file参数，指定一个密码文件来解决
echo '123456' > /etc/rsync.password
chmod 600 /etc/rsync.password     # 权限必须600，否则报错
rsync -avz --password-file=/etc/rsync.password /export/data/logs/ backup_user@node2::node1_logs
```

☁ 实现同步脚本优化

代码块

```
# 编写inotify监控脚本
```

```
vim /root/real_time_sync.sh

添加以下内容：是基于RSYNC服务的脚本文件：
#!/bin/bash
# 定义源目录和目标目录（目标目录是远程服务器的目录）
SOURCE_DIR="/export/data/logs"
#TARGET_DIR="root@192.168.88.102:/export/data/node1_exe_backup_log"  # 替换为目标服务器的用户名和目录
# 使用RSYNC服务进行数据同步
TARGET_DIR="backup_user@192.168.88.102::node1_logs"

# 日志文件
LOG_FILE="/var/log/rsync_realtime_sync.log"

# 使用inotifywait监听${WATCH_DIR}目录中的文件变化
inotifywait -m -r -e modify,create,delete,move "${SOURCE_DIR}" | while read path action file;
do
    echo "Detected ${action} on ${file} in ${path}. Starting rsync..." >> "$LOG_FILE"
    # 执行rsync同步操作
    rsync -avz --port=873 --delete --password-file=/etc/rsync.password "${SOURCE_DIR}/" "${TARGET_DIR}"  >> "$LOG_FILE" 2>&1
    echo "Sync completed for ${file} in ${path}." >> "$LOG_FILE"
done
```