任务背景

某公司为了保证开发人员线上代码的安全性，现需要对开发人员的代码进行备份。

任务要求

1. 备份机器需要每天凌晨1:03分定时同步MIS服务器的/app/java_project目录下的所有文件。
2. 要求记录同步日志，方便同步失败分析原因。（不仅仅进行同步，还要求有同步日志）

任务拆解

1. 选择合适的备份工具和方法来备份（scp可以实现，不是很完美——>rsync）
2. 掌握所选择工具的用法
3. 编写脚本（超纲）让计划任务去执行

涉及知识点

- ==rsync命令使用==（新知识点）
- ==rsync作为后台程序如何使用==（新知识点）
- crontab计划任务（旧知识点）

课程目标

- 能够使用rsync命令实现==本地==文件同步(类似cp)
- 能够使用rsync命令实现==远程==文件同步（类似scp）
- 能够使用rsync作为后台程序进行数据同步

理论储备

一、rsync介绍

- rsync的好姐妹
    - sync 同步：刷新文件系统缓存，强制将修改过的数据块写入磁盘，并且更新超级块。

    - async 异步：将数据先放到缓冲区，再周期性（一般是30s）的去同步到磁盘。
    - rsync 远程同步：remote synchronous
- rsync的特点
    - 可以镜像保存整个目录树和文件系统
    - 可以保留原有的权限(permission,mode)，owner,group,时间(修改时间,modify time)，软硬链接，文件acl，文件属性(attributes)信息等
    - 传输==效率高==，使用同步算法，只比较变化的（增量备份）

      file1.txt file2.txt file3.txt(A服务器)

      rsync实现数据同步 => 只同步file3.txt => 增量备份

      file1.txt file2.txt(B服务器)
    - 支持匿名传输，方便网站镜像；也可以做验证，加强安全

rsync与scp区别?

```
# 两者都可以实现远程同步，但是相对比而言，rsync能力更强
① 支持增量备份
② 数据同步时保持文件的原有属性
```

二、rsync的语法

- man rsync

```
NAME
       rsync - a fast, versatile, remote (and local) file-copying tool
       //一种快速、通用、远程（和本地）的文件复制工具

SYNOPSIS
       Local:  rsync [OPTION...] SRC... [DEST]

       Access via remote shell:
       //通过远程shell访问（命令）
         Pull: rsync [OPTION...] [USER@]HOST:SRC... [DEST]
         Push: rsync [OPTION...] SRC... [USER@]HOST:DEST

       Access via rsync daemon:
       //通过后台程序访问（作为服务）
         Pull: rsync [OPTION...] [USER@]HOST::SRC... [DEST]
               rsync [OPTION...] rsync://[USER@]HOST[:PORT]/SRC... [DEST]
         Push: rsync [OPTION...] SRC... [USER@]HOST::DEST
               rsync [OPTION...] SRC... rsync://[USER@]HOST[:PORT]/DEST
```

- rsync命令参数

```
    -v      详细模式输出
    -a      归档模式，递归的方式传输文件，并保持文件的属性，equals -rlptgoD
    -r      递归拷贝目录
    -l      保留软链接
    -p      保留原有权限
    -t      保留原有时间（修改）
    -g      保留属组权限
    -o      保留属主权限
```

```
    -D      等于--devices  --specials    表示支持b,c,s,p类型的文件
    -R      保留相对路径
    -H      保留硬链接
    -A      保留ACL策略
    -e      指定要执行的远程shell命令
    -E      保留可执行权限
    -X      保留扩展属性信息   a属性
```

三、rsync命令使用

1. 本机同步

```
注意：
1. 本地数据同步的时候，源目录后面的“/”会影响同步的结果
     # rsync -av /dir1/ /dir3       //只同步目录下面的文件到指定的路径
     # rsync -av /dir1 /dir2        //将当前目录dir1和目录下的所有文件一起同步

2. -R：不管加不加"/"，都会将源数据的绝对路径一起同步
    # rsync -avR /dir1/ /dir2/
3. --delete：删除目标目录里多余的文件
    # rsync -avR --delete /dir1/ /dir2/
```

2. 远程同步

```
pull: rsync -av user@host:/path local/path
# rsync -av root@10.1.1.1:/etc/hosts /dir1/
# rsync -av root@10.1.1.1:/backup /dir1
push: rsync -av local/path user@host:/path
# rsync -av /dir1/aa1 code@10.1.1.1:/home/code
```

思考：

rsync远程同步数据时，默认情况下为什么需要密码？如果不想要密码同步怎么实现？

```
因为两台Linux服务器在连接时，默认使用的还是SSH协议。由于没有做免密登录，所以还是需要输入对方服务器的密码。
如果不想输入密码，可以使用免密登录来实现。
# JumpServer操作
# ssh-keygen
# ssh-copy-id root@10.1.1.250
```

四、rsync作为服务使用

思路：

对外提供服务——>端口监听——>==启动服务==——>启动脚本（没有）——>配置文件（修改需求）

1. 启动它（找启动脚本—>没找到）
2. 寻求帮助

```
man 1 rsync
man 5 rsyncd.conf
rsync --help
Use "rsync --daemon --help" to see the daemon-mode command-line options.
结论：
rsync后台程序去启动，那么rsync --daemon [options]
```

3. 尝试启动rsync程序

```
[root@jumper ~]# rsync --daemon
Failed to parse config file: /etc/rsyncd.conf
说明：先尝试以后台程序的方式启动它
原因：没有配置文件（默认没有）
解决：创建配置文件
[root@jumper ~]# touch /etc/rsyncd.conf
[root@jumper ~]# rsync --daemon
[root@jumper ~]# ps -ef|grep rsync
root       3814      1  0 11:43 ?        00:00:00 rsync --daemon
root       3817   2826  0 11:44 pts/0    00:00:00 grep rsync
[root@jumper ~]# netstat -nltup|grep rsync
tcp        0      0 0.0.0.0:873                 0.0.0.0:*
LISTEN      3814/rsync
tcp        0      0 :::873                      :::*
 LISTEN      3814/rsync
结论：
1. rsync启动时必须要读取配置文件，如果没有报错
2. rsync默认情况下，端口是873，协议tcp
```

4. 了解rsync的配置文件

```
man 5 rsyncd.conf
全局的参数
    port = xxx
    pid file = xxx
    。。。
局部模块
[模块名1]
    path = /dir1
    uid = xxx
    gid = xxx
    log file = xxx
    max connections =4
[模块名2]
    path = /dir2
    uid =xxx
    gid = xx
```

任务解决方案

任务分析

```
1. 备份的本质
    文件的拷贝，cp scp rsync
2. 选择工具
    rsync:
    1)作为服务不需要交互
    2)可以实现增量拷贝
3. 掌握工具的使用
    命令：
    服务：编写配置文件/etc/rsyncd.conf-->rsync --daemon-->873
```

实施步骤

```
环境准备：
code:10.1.1.10
backup:10.1.1.100

一、以下操作在server（10.1.1.1）服务端完成
1. 在线上环境(server)创建相应的目录
[root@server ~]# mkdir /app/java_project -p
以下数据是为了测试用：
[root@server ~]# touch /app/java_project/file{1..10}.java
[root@server ~]# mkdir /app/java_project/aa{1..3}
2. 在线上环境中将rsync作为后台程序运行
    1)创建配置文件
    [root@server ~]# cat /etc/rsyncd.conf
    [app]
    path = /app/java_project
    log file = /var/log/rsync.log
    2)启动rsync服务
    rsync --daemon
    3)验证服务成功启动
    [root@server ~]# netstat -nltp|grep 873

二、备份机器backup(10.1.1.250)操作
1. 编写脚本来同步文件
[root@backup ~]# mkdir /backup/app1_java -p
[root@backup ~]# rsync -a root@10.1.1.1::    查看服务端的模块名字
app
[root@backup ~]# cat rsync_java.sh
#!/bin/bash
rsync -av root@10.1.1.1::app /backup/app1_java/ &>/dev/null

[root@backup ~]# chmod +x rsync_java.sh
2. 交给计划任务去执行
[root@backup ~]# crontab -e
[root@backup ~]# crontab -l
03 01 * * * /root/rsync_java.sh
3. 测试验证
1)通过修改系统时间验证
[root@backup ~]# date -s "2018-12-29 01:02:50"
Sat Dec 29 01:02:50 CST 2018
2)查看计划人任务日志
[root@backup ~]# tail -f /var/log/cron
Dec 29 01:03:17 MissHou CROND[4410]: (root) CMD (/root/rsync_java.sh)
```

```
3)查看数据文件是否同步过来
[root@backup ~]# ll /backup/app1_java/
total 12
drwxr-xr-x 2 root root 4096 Dec 28 15:14 aa1
drwxr-xr-x 2 root root 4096 Dec 28 15:14 aa2
drwxr-xr-x 2 root root 4096 Dec 28 15:14 aa3
-rw-r--r-- 1 root root    0 Dec 28 15:14 file10.java
-rw-r--r-- 1 root root    0 Dec 28 15:14 file1.java
-rw-r--r-- 1 root root    0 Dec 28 15:14 file2.java
-rw-r--r-- 1 root root    0 Dec 28 15:14 file3.java
-rw-r--r-- 1 root root    0 Dec 28 15:14 file4.java
-rw-r--r-- 1 root root    0 Dec 28 15:14 file5.java
-rw-r--r-- 1 root root    0 Dec 28 15:14 file6.java
-rw-r--r-- 1 root root    0 Dec 28 15:14 file7.java
-rw-r--r-- 1 root root    0 Dec 28 15:14 file8.java
-rw-r--r-- 1 root root    0 Dec 28 15:14 file9.java
```

任务总结

1. rsync干嘛用的?
2. rsync特点：增量同步、保留文件的属性信息
3. rsync服务好处，不需要密码，873/tcp，对服务本身做一些限制
4. 该任务所选择方法有很多种：
   1） 配置免密码登录，然后将命令写到脚本里，交给计划任务完成，缺陷：每次全量拷贝
   2） ==将rsync做成后台程序（服务），然后将命令写到脚本里，交给计划任务完成==

课后扩展

==**思考：如果为了考虑安全性，既想要使用后台服务运行，又想要密码如何实现？**==

1. 需要密码同步

环境

backup：10.1.1.250　　/backup

app1-server：10.1.1.1　　/app/java_project

① 修改主配置文件

```
vim  /etc/rsyncd.conf
motd file=/etc/rsyncd.welcome            消息文件
[app1]
comment=共享给客户端看到的名字，可以自己定义
path=/app/java_project
auth users = user1,user2
secrets file = /etc/rsyncd.secrets
```

② 创建安全用户，该文件不能被其他人查看

```
[root@app1-server ~]# vim /etc/rsyncd.secrets
user1:123
user2:123
[root@app1-server ~]# ll /etc/rsyncd.secrets
-rw-r--r-- 1 root root 20 Aug 29 09:57 /etc/rsyncd.secrets
[root@app1-server ~]# chmod 600 /etc/rsyncd.secrets

注意：密码文件的名字必须和配置文件里定义的保持一致

创建一个消息文件
[root@app1-server ~]# echo "welcome to itcast 哈哈哈" > /etc/rsyncd.welcome
```

③ 测试验证

```
[root@app1-server ~]# ps -ef|grep rsync
root       2962      1  0 09:48 ?        00:00:00 rsync --daemon
root       3034   2534  0 09:59 pts/0    00:00:00 grep rsync
[root@app1-server ~]# pkill -9 rsync
[root@app1-server ~]# ps -ef|grep rsync
root       3038   2534  0 09:59 pts/0    00:00:00 grep rsync

[root@app1-server ~]# rsync --daemon

在backup服务器上测试：
[root@backup ~]# rsync -av user1@10.1.1.1::app1 /backup/
welcome to itcast 哈哈哈

Password:
receiving incremental file list
./
aa1/
aa2/
aa3/

sent 70 bytes  received 259 bytes  13.43 bytes/sec
total size is 0  speedup is 0.00

注意：
user1用户是服务器端app1-server上的密码文件里定义的用户；不是当前操作系统的用户。
```

说明：
服务端rsync服务是以什么用户运行，则必须保证secrets file文件拥有者必须是同一个;

假设root运行rsync --daemon，则secrets file的owner也必须是root；secrets file权限必须是600

总结：

- 没有密码有好处也有坏处，好处是不需要密码方便写脚本做远程同步；坏处就是不安全，但你可以使用防火墙等来加强安全。
- 同步可能出现的问题:
  - 如果同步报permission denied这种，可能是服务端selinux没有关闭
  - 同步时间慢:
    - 解决方法:绑定对方主机名
- 如果你希望有密码，可以用rsyncd本身自带的secrets file来做用户验证

2. rsync+inotify实时同步

需求1：将主机A上的/dir1目录的数据实时同步到A主机（本机）的/dir2目录里

```
1. 安装inotify软件
# tar xf inotify-tools-3.13.tar.gz -C /usr/src/
# cd /usr/src/inotify-tools-3.13/
# ./configure
# make
# make install

安装完后，就会产生下面两个命令
/usr/local/bin/inotifywait      等待
/usr/local/bin/inotifywatch     看守

2. 编写脚本来同步
#!/bin/bash
/usr/local/bin/inotifywait -mrq -e modify,delete,create,attrib,move /dir1 |while
read events
        do
                rsync -a --delete /dir1 /dir2/
                echo "`date +%F\ %T`出现事件$events" >> /var/log/rsync.log 2>&1
        done

# chmod +x 1.sh
# nohup ./1.sh  &

--delete 参数：如果目标目录里的数据比源目标多，那么需要删除无用的相关的数据

3. 测试验证


如有如下错误：
@ERROR: chroot failed           //rsyncd.conf文件中path路径不存在导致
rsync error: error starting client-server protocol (code 5) at main.c(1503)
[sender=3.0.6]
```

需求2:

app1-server服务器上的/app/java_project目录的文件和backup主机上的/backup目录实时同步

```
环境：
app1-server  10.1.1.1
backup 10.1.1.250
分析：
rsync本身不可以做到数据的实时同步，需要借助第三方工具，intotify工具。实现线上环境目录发生改变
立马同步到backup主机，是单向同步。

步骤：
1. 在app1-server上安装inotify工具
将软件包上传到虚拟主机上（app1-server）
[root@app1-server ~]# ls /soft/
inotify-tools-3.13.tar.gz
[root@app1-server ~]# tar xf inotify-tools-3.13.tar.gz
[root@app1-server ~]# ls /soft/
inotify-tools-3.13  inotify-tools-3.13.tar.gz
```

```
[root@app1-server ~]# cd inotify-tools-3.13

[root@app1-server ~]# ./configure
[root@app1-server ~]# make
[root@app1-server ~]# make install
[root@app1-server ~]# ll /usr/local/bin/
total 80
-rwxr-xr-x 1 root root 38582 Aug 29 10:48 inotifywait
-rwxr-xr-x 1 root root 40353 Aug 29 10:48 inotifywatch
```

2. 查看命令如何使用，然后编写脚本来实现目录的监控
注意：该脚本应该在app1-server上运行

```
[root@app1-server ~]# inotifywait --help
-m 保持监控状态
-r 递归监控
-q 只打印事件
-e 指定事件
```

事件：

```
move    移动
delete  删除
create  创建
modify  修改
attrib  属性信息
```

编写脚本实时监控/app/java_project目录

```
[root@app1-server ~]# echo hello
hello
[root@app1-server ~]# echo hello|while read i;do echo $i;done
hello
```

解释：上一条命令执行的结果hello赋值给i变量，然后打印出i变量的值。

```
vim /root/1.sh
#!/bin/bash
/usr/local/bin/inotifywait -mrq -e modify,delete,create,attrib,move
/app/java_project |while read events
        do
                rsync -a --delete /app/java_project/ 10.1.1.250:/backup
                echo "`date +%F\ %T`出现事件$events" >> /var/log/rsync.log 2>&1
        done

# chmod +x 1.sh      增加可执行权限
# ./1.sh  &          将脚本放到后台去运行
```

注意：
如果单纯使用命令去推的话，正常情况下需要密码，不利于脚本编写，在这里提供2中解决方案：
1）设置免密码登录
2）在backup服务器上将rsync作为后台程序运行

3. 测试验证
app1-server上操作目录：

```
[root@app1-server java_project]# mkdir bbb{1..3}
[root@app1-server java_project]# rm -f file{1..3}

tail -f /var/log/rsync.log
2018-08-29 11:13:35出现事件/app/java_project/ CREATE,ISDIR bbb1
2018-08-29 11:13:35出现事件/app/java_project/ CREATE,ISDIR bbb2
```

```
2018-08-29 11:13:35出现事件/app/java_project/ CREATE,ISDIR bbb3
2018-08-29 11:14:15出现事件/app/java_project/ DELETE file1
2018-08-29 11:14:15出现事件/app/java_project/ DELETE file2
2018-08-29 11:14:15出现事件/app/java_project/ DELETE file3
```

3. rsync托管xinetd

独立服务：独立启动脚本 ssh ftp nfs dns ...

依赖服务：没有独立的启动脚本 rsync telnet 依赖xinetd服务（独立服务）

1. 平时不占用系统的运行资源
2. xinetd服务管理依赖服务
3. 一些轻量级服务会托管给xinetd服务

如何将rsync托管给xinetd服务去管理?

```
1. 安装相应的软件
[root@jumper ~]# yum -y install xinetd

2. 查看软件类别
[root@jumper ~]# rpm -ql xinetd
/etc/rc.d/init.d/xinetd
/etc/xinetd.conf
/etc/xinetd.d   xinetd服务管理的所有轻量级服务的目录
/usr/sbin/xinetd
/usr/share/man/man5/xinetd.conf.5.gz

3. 查看rsync软件列表
[root@jumper ~]# rpm -ql rsync
/etc/xinetd.d/rsync
/usr/bin/rsync

4. 了解配置文件
1）xinetd服务的配置文件
man 5 xinetd.conf
only_from       只允许访问
no_access       拒绝访问
access_times    控制访问服务的时间段
log_type            指定日志类型
interface       并发连接数
per_source      每个IP的最大连接数


2）rsync配置文件 /etc/xinetd.d/rsync

service rsync
{
    disable = no        //开关；no表示开启该服务；yes表示关闭服务
    flags       = IPv6
    socket_type     = stream
    wait            = no
    user            = root
    server          = /usr/bin/rsync
    server_args     = --daemon
    log_on_failure  += USERID
```

```
}

5. 把rsync服务的开关打开
6. 启动xinetd服务
[root@jumper ~]# service xinetd restart
Stopping xinetd:                                           [  OK  ]
Starting xinetd:                                           [  OK  ]
[root@jumper ~]# netstat -nltp|grep 873
tcp        0      0 :::873                      :::*
 LISTEN      6209/xinetd

到此，rsync服务就托管给了xinetd服务管理；后续需要有些需求，修改配置文件：
/etc/rsyncd.conf
/etc/xinetd.d/rsync
```

课后实战

将主机A上的/share/dir1目录远程同步到主机B上的/backup/dir2目录里，要求如下：

1、把日志记录到/var/log/rsyncd.log

2、共享模块要求隐藏（也就是说客户端查看不到这个模块名）

3、并且同时只能1个客户端连接进行同步这个module

4、只能允许x.x.x.x(ip你自定义)同步这个module