Journal of Artificial Intelligence and Engineering Applications

Website: https://ioinformatic.org/

June 2026. Vol. 5. No. 3; e-ISSN: 2808-4519

Implementing Automatic Incremental Backups Using Rsync on Debian 12

Teresa Martuah Purba[1]*, Angela Steffani Br Sitanggang[2], Lotar Mateus Sinaga[3]

[1,2,3]*Santo Thomas Catholic University*

teresamartuahpurba26@gmail.com[1*], angelasteffani2005@gmail.com[2], lotarmateus88@gmail.com[3]

Abstract

Data is a critical asset whose availability and security must be safeguarded within an information system. Data loss can occur due to hardware failure, user error, or system failure, all of which can disrupt an organization's operations. Therefore, a data backup mechanism capable of ensuring the availability of information is necessary.This study aims to implement an automatic incremental backup system using rsync on the Debian 12 operating system. The system is built using two servers—a primary server and a backup server—connected via a network. Data synchronization is performed using rsync over the SSH protocol, while the backup automation process is managed using cron.Test results show that rsync is capable of synchronizing data efficiently by transferring only files that have changed. Additionally, cron allows the backup process to run automatically according to a predetermined schedule. The system developed can enhance data security, simplify the backup process, and reduce the risk of data loss on the server.

Keywords: *Semicolon Backup server; Cron; Debian 12; Incremental backup; Rsync.*

1. Introduction

Advances in information technology have made the need for data security and availability increasingly important. Data is a highly valuable asset for both organizations and individuals because it is used to support various activities and information processing. Data loss can result in system operational disruptions, information corruption, and significant financial losses [1].

Data disruptions can be caused by various factors, such as hardware failure, user error, operating system failure, or cyberattacks. Therefore, a data protection mechanism is needed to ensure the availability of information and minimize the risk of data loss [10].

One method used to ensure data security is implementing a backup system. Backup is the process of creating copies of data to restore it in the event of damage or loss to the primary system. Implementing server backups can improve system reliability and maintain service continuity when the primary server experiences an outage [8].

The incremental backup method is an efficient backup technique because it transfers only the files that have changed since the previous backup. This technique reduces the use of storage media and network bandwidth compared to the full backup method [1]. In addition, modern backup concepts also emphasize the importance of storing multiple copies of data to improve resilience against data loss [9].

On the Linux operating system, the rsync utility is one of the most widely used applications for data synchronization. Rsync is capable of transferring files efficiently by sending only the changes that have occurred in the files, thereby speeding up the data backup process [11]. The backup process can also be run automatically using cron, so administrators do not need to perform backups manually [17].

This study aims to implement an automated incremental backup system using rsync on the Debian 12 operating system. The system is built using two servers connected via a network: a primary server and a backup server. It is expected that the implemented system will enhance data security, simplify the backup process, and reduce the risk of data loss on the servers.

2. Research Methodology

2.1. Research

The research method used in this study is the system implementation and testing method. The research stages began with system requirements analysis, network configuration, software installation, and backup system implementation, and concluded with testing the backup results on two Debian 12 servers. This method aims to determine the performance of an incremental backup system using rsync to

automatically synchronize data on Linux servers. Testing was conducted to ensure that the backup process runs smoothly and that data is successfully copied to the backup server.

2.2. Hardware

The hardware used in this study served as a platform for implementing and testing the backup system. The equipment consisted of two laptops running Debian 12 virtual machines as the primary server and the backup server. The hardware specifications are shown in the following table.

Table 1: Hardware Specifications

No	Devices	Functions
1	Laptop 1	Primary Server
2	Laptop 2	Backup Server
3	Wi-Fi network	Communication Media
4	VirtualBox	Server Virtualization

2.3. Software

The software used in this study serves to support the implementation of an automated backup system. This software includes an operating system, communication services, backup utilities, and virtualization applications. The list of software used is shown in the following table.

Table 2: Software

No	Software	Functions
1	Debian 12	Server operating system
2	OpenSSH Server	Remote server
3	Rsync	Data synchronization
4	Cron	Backup scheduling
5	VirtualBox	Virtualization

2.4. Network Topology

The network topology consists of two Debian 12 virtual machines running on two different laptops. The first server serves as the primary server that stores the original data, while the second server serves as the backup server that receives the synchronized data. The IP address configuration is shown in the following table.

Table 3: Network Configuration

Server	Hostname	IP Address	Functions
Primary Server	poerba	172.16.0.90	Data Sources
Backup Server	fanizer	172.16.1.164	Backup Server

2.5. Research Phases

The research stages were carried out systematically, from network configuration to testing the backup results. The purpose of this research process was to ensure that the backup system developed would function properly and align with the research objectives. The flow of the research stages is shown in the following figure.

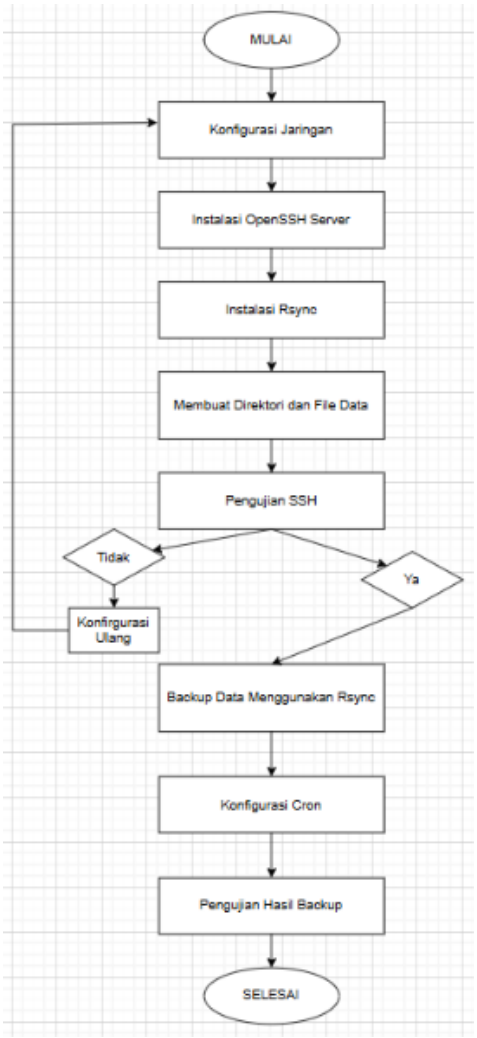

Fig. 1: Research Flowchart

2.6. Figures and tables

2.6.1. Installing OpenSSH Server

OpenSSH is used to establish communication between the primary server and the backup server.
The installation command used is:
apt update
apt install openssh-server -y
The SSH service is then started using the command:
systemctl enable ssh
systemctl start ssh

2.6.2. Installing Rsync

Rsync is used as a utility for synchronizing data between servers.
Installation command:
apt install rsync -y

2.6.3. Creating the Data Directory

On the main server, create the data directory to be backed up.
mkdir /data
Next, create several test files.
echo “Student data” > students.txt
echo “Faculty data” > faculty.txt
echo “Journal data” > journal.txt

2.6.4. Creating a Backup Directory

A destination directory for the backup is created on the backup server.
mkdir -p /backup/data

2.6.5. Testing the SSH Connection

This test is performed to ensure that the two servers can connect to each other.
ssh root@172.16.1.164
If the login is successful, communication between the servers is working properly.

2.6.6. Implementing a Backup Using Rsync

The backup process is performed using the following command:
rsync -av /data/ root@172.16.1.164:/backup/data/
This command will copy all files in the /data directory to the backup server via the SSH protocol.

2.6.7. Automating Backups Using Cron

First, create a backup script: *nano /script/backup.sh*
Script contents: *rsync -av /data/ root@172.16.1.164:/backup/data/*
Grant execution permissions: *chmod +x /script/backup.sh*
Add a cron job: *crontab -e*
Contents: */5 * * * * /script/backup.sh
This configuration causes the backup process to run automatically every five minutes.

3. Results and Discussion

In this study, an automatic incremental backup system using rsync was implemented on the Debian 12 operating system. Testing was conducted using two servers—one serving as the primary server and the other as the backup server—which were connected via a network.

3.1. Network Connection Testing

Network connection testing was performed to ensure that the two servers could communicate with each other before the backup process was executed. The test was performed using the ping command from the primary server to the backup server.
Ping 172.16.1.164
The test results showed that data packets were successfully sent and received without any packet loss, indicating that communication between the servers was functioning properly.

3.2. SSH Connection Test

After the network connection was established, an SSH service test was performed to ensure the primary server could remotely access the backup server.
Command used: *ssh root@172.16.1.164*
After entering the correct password, the primary server successfully logged into the backup server. This indicates that the OpenSSH service is functioning properly on both servers.

```
root@poerba:~# ping 172.16.1.164
PING 172.16.1.164 (172.16.1.164) 56(84) bytes of data.
64 bytes from 172.16.1.164: icmp_seq=1 ttl=64 time=102 ms
64 bytes from 172.16.1.164: icmp_seq=2 ttl=64 time=8.10 ms
64 bytes from 172.16.1.164: icmp_seq=3 ttl=64 time=7.20 ms
64 bytes from 172.16.1.164: icmp_seq=4 ttl=64 time=7.01 ms
64 bytes from 172.16.1.164: icmp_seq=5 ttl=64 time=8.56 ms
64 bytes from 172.16.1.164: icmp_seq=6 ttl=64 time=6.54 ms
^C
--- 172.16.1.164 ping statistics ---
6 packets transmitted, 6 received, 0% packet loss, time 5017ms
rtt min/avg/max/mdev = 6.538/23.254/102.114/35.273 ms
```

Fig. 2: Results of the Inter-Server Ping Test

3.3. Creating Test Data

Test data was created in the /data directory on the main server. This data was used to verify the success of the synchronization process using rsync.
Commands used:

cd /data
ls -l
The test results show that there are several files to be backed up, namely:
dosen.txt , jurnal.txt , mahasiswa.txt

```
root@poerba:~# mkdir -p /data
root@poerba:~# cd /data
root@poerba:/data# ls -l
total 12
-rw-r--r-- 1 root root 11 Jun  7 16:37 dosen.txt
-rw-r--r-- 1 root root 12 Jun  7 16:38 jurnal.txt
-rw-r--r-- 1 root root 15 Jun  7 16:36 mahasiswa.txt
root@poerba:/data# ssh root@172.16.1.164
root@172.16.1.164's password:
Linux fanizer 6.1.0-49-amd64 #1 SMP PREEMPT_DYNAMIC Debian 6.1.174-1 (2026-05-26) x86_64
```

Fig. 4: Test Data on the Main Server

3.4. Implementing a Backup Using Rsync

The backup process was performed using the rsync utility run on the main server. The command used was:
rsync -av /data/ root@172.16.1.164:/backup/data/
The -a parameter is used to preserve file attributes, while the -v parameter is used to display backup process information.
Test results show that all files were successfully copied to the backup server.

```
root@poerba:~# rsync -av /data/ root@172.16.1.164:/backup/data/
root@172.16.1.164's password:
sending incremental file list
./
dosen.txt
jurnal.txt
mahasiswa.txt

sent 302 bytes  received 76 bytes  36.00 bytes/sec
total size is 38  speedup is 0.10
```

Fig. 5: Backup Process Using Rsync

3.5. Testing the Backup Results

After the backup process was completed, a test was conducted on the backup server by examining the contents of the destination directory.
The command used was: *ls -l /backup/data*
Test results show that all files on the main server were successfully copied to the backup server.

```
root@fanizer:~# ls /backup/data
backup.txt  dosen.txt  jurnal.txt  mahasiswa.txt  otomatis.txt
```

Fig 6. Backup Results on the Backup Server

3.6. Cron Testing

Backup automation is performed using cron so that the backup process can run without administrator intervention.
Cron configuration is performed using the command: *crontab -e*
The configuration used is: **/5 * * * * /script/backup.sh*
This configuration causes the backup script to run every five minutes.

```
root@poerba:~# crontab -l
*/5 * * * * /script/backup.sh
# Edit this file to introduce tasks to be run by cron.
#
# Each task to run has to be defined through a single line
# indicating with different fields when the task will be run
# and what command to run for the task
#
# To define the time you can provide concrete values for
# minute (m), hour (h), day of month (dom), month (mon),
# and day of week (dow) or use '*' in these fields (for 'any').
#
# Notice that tasks will be started based on the cron's system
# daemon's notion of time and timezones.
#
# Output of the crontab jobs (including errors) is sent through
# email to the user the crontab file belongs to (unless redirected).
#
# For example, you can run a backup of all your user accounts
# at 5 a.m every week with:
# 0 5 * * 1 tar -zcf /var/backups/home.tgz /home/
#
# For more information see the manual pages of crontab(5) and cron(8)
#
# m h  dom mon dow   command
```

Fig. 7: Cron Configuration

3.7. Cron Testing

Based on the test results, the incremental backup system using rsync was successfully implemented on Debian 12. The data synchronization process can run over the network using the SSH protocol.
The use of rsync allows the backup process to be performed efficiently because it only transfers files that have changed. Additionally, the use of cron automates the backup process, eliminating the need for administrators to perform backups manually.
The implementation also demonstrates that using a backup server can improve data availability and reduce the risk of data loss in the event of a failure on the primary server. These findings align with the research by Santos and Bernardino, who state that rsync is one of the most efficient backup solutions for Linux systems [1].

The backup process is performed using the rsync utility, which is run on the main server. The command used is:
rsync -av /data/ root@172.16.1.164:/backup/data/
The -a parameter is used to preserve file attributes, while the -v parameter is used to display backup process information.
Test results show that all files were successfully copied to the backup server.

```
root@poerba:~# rsync -av /data/ root@172.16.1.164:/backup/data/
root@172.16.1.164's password:
sending incremental file list
./
dosen.txt
jurnal.txt
mahasiswa.txt

sent 302 bytes  received 76 bytes  36.00 bytes/sec
total size is 38  speedup is 0.10
```

Fig. 8: Backup Process Using Rsync

4. Conclusion

Based on the results of the implementation and testing conducted, it can be concluded that the automatic incremental backup system using rsync on the Debian 12 operating system was successfully implemented. The connection between servers can be established using the OpenSSH service, allowing the data synchronization process to be performed over the network.
The use of rsync enables efficient data backup because it transfers only files that have changed. Additionally, the use of cron allows the backup process to run automatically according to a predetermined schedule without requiring administrator intervention.
Test results show that all files on the primary server were successfully copied to the backup server. Thus, the system developed enhances data security, simplifies the backup process, and reduces the risk of data loss due to system failure or data loss on the primary server.

References

[1] A. Santos and J. Bernardino, “Open source tools for remote incremental backups on Linux: An experimental evaluation,” J. Syst. Integr., vol. 5, no. 3, pp. 3–13, 2014, doi: 10.20470/jsi.v5i3.205.
[2] D. Meister and A. Brinkmann, “RevDedup: A reverse deduplication storage system optimized for reads to latest backups,” in Proc. 11th USENIX Conf. File Storage Technol., pp. 281–294, 2013.
[3] A. Arshad, “TCP server fault tolerance using connection migration to a backup server,” Univ. Warwick, Tech. Rep., 2009.
[4] A. Dani, S. Mangade, P. Nimbalkar, and H. Shirwadkar, “Next4: Snapshots in Ext4 file system,” arXiv:2403.06790, 2024.
[5] J. Mohan, R. Kadekodi, and V. Chidambaram, “Analyzing IO amplification in Linux file systems,” arXiv:1707.08514, 2017.

[6] M. Jia, E. H. M. Sha, Q. Zhuge, R. Xu, and S. Gu, "Rapid recovery of program execution under power failures for embedded systems with non-volatile memory," arXiv:2209.08826, 2022.
[7] M. Nikolic and D. Olteanu, "Incremental view maintenance with triple lock factorization benefits," arXiv:1703.07484, 2017.
[8] D. H. Yudhistira and A. Prasetyo, "The design and implementation of backup server using the failover method," J. Multimedia Comput. Sci., 2025.
[9] J. Lewis, "Evolving the 3-2-1 backup rule for more resilient data," J. eScience Librarianship, vol. 13, no. 1, 2024.
[10] A. Kumar et al., "Artificial intelligence driven approach for securing backup data and enhancing cyber resilience in sustainable smart infrastructure," Sci. Rep., 2026.
[11] M. Polivakha, "Perform incremental backups in Linux using rsync," 2024.
[12] Red Hat, "Incremental backups," Red Hat Enterprise Linux Documentation.
[13] IBM, "Journal-based backup on Linux systems," IBM Storage Protect Documentation.
[14] IBM, "Nojournal option for incremental backup," IBM Storage Protect Documentation.
[15] "Build a home terabyte backup system using Linux," Linux J., 2005.
[16] "An automated reliable backup solution," Linux J., 2006.
[17] "Automating remote backups," Linux J., 2010.