# Secure Remote Backup System for Small Businesses

This wiki documents the implementation of a secure remote backup system designed for small businesses with limited IT resources.

## Quick Navigation

## Project Information

**Student:** Buys Enock Onkarabile **Student Number:** 219013044 **Module:** Data Communications – Linux System Administration (IT28X97)

# Secure Remote Backup System for Small Businesses

## Project Overview

This project addresses a critical challenge faced by small businesses that rely on personal computers or small servers to store important business files. Many small businesses lack robust data backup systems due to financial constraints, limited resources, and insufficient IT expertise. The solution provides an easy-to-use, affordable backup system using Linux that offers:

- Automated local and remote backups
- Strong encryption for data protection
- Data integrity verification
- Simple web interface for monitoring and restoration
- No need for advanced technical skills

# System Architecture

The system consists of two main components:

Backup Server (hostname: backup-server): Centralized storage for all backups with IP 192.168.56.10

Client System (hostname: client-system): Business computer requiring backup protection with IP 192.168.56.20

Both systems run Ubuntu 22.04 LTS with bridged networking to ensure internet connectivity.

## Core Technologies

leveraging several proven open-source technologies:

- BorgBackup: Deduplicating backup software providing compression and encryption
- Borgmatic: Configuration-driven automation wrapper for BorgBackup
- Nginx: Web server for hosting the backup management interface
- Flask: Python web framework for creating the management dashboard
- Docker: Container platform for isolation and simplified deployment
- Prometheus & Grafana: Monitoring and visualization tools
- Postfix: Email server for backup notifications
- Rclone: Tool for synchronizing with cloud storage services

# Installation Process

## Prerequisites

Two machines (physical or virtual) with Ubuntu 22.04 LTS installed Network connectivity between both systems Internet access for package installation and cloud backup (if configured) Basic understanding of Linux command line operations

## 1. Initial Setup for Both Systems

### 1.1 Base System Installation

Install Ubuntu 22.04 LTS on both VMs with the following configurations: For Backup Server:

Hostname: backup-server Username: backup-admin Password: Choose a strong password (different from encryption passphrase)

For Client System:

Hostname: client-system Username: business-user Password: Choose a strong password

### 1.2 Update Systems

On both systems, run:

```
sudo apt update
sudo apt upgrade -y
```

### 1.3 Configure Hostnames and Hosts File

On both systems, edit the hosts file:

```
sudo nano /etc/hosts
```

Add the following entries to both systems' hosts files:

```
192.168.1.10    backup-server
192.168.1.20    client-system
```

### 1.4 Test Connectivity

Verify communication between systems:

```
# From client-system
ping -c 4 backup-server
From backup-server
ping -c 4 client-system
```

## 2. Backup Server Configuration

### 2.1 Install Required Packages

```
sudo apt install -y openssh-server nginx python3-pip python3-dev libssl-dev
libffi-dev docker.io docker-compose certbot python3-certbot-nginx prometheus
prometheus-node-exporter grafana postfix
```

### 2.2 Configure SSH

Edit the SSH configuration file:

```
sudo nano /etc/ssh/sshd_config
```

Make these security-enhancing changes:

```
PermitRootLogin no
PasswordAuthentication no
```

Restart SSH to apply changes:

```
sudo systemctl restart sshd
```

### 2.3 Create Backup Storage Directory

```
sudo mkdir -p /backup/repositories
sudo chmod 700 /backup/repositories
sudo chown backup-admin:backup-admin /backup/repositories
```

### 2.4 Install Borg Server

```
sudo apt install -y borgbackup
```

### 2.5 Configure Docker

Docker should be installed from step 2.1. Enable and start the service:

```
sudo systemctl enable docker
sudo systemctl start docker
sudo usermod -aG docker backup-admin
```

### 2.6 Configure Postfix for Email Notifications

During Postfix installation, choose “Internet Site” and enter your server's FQDN (you can use localhost for testing). Edit the Postfix configuration:

```
sudo nano /etc/postfix/main.cf
```

Update these settings:

```
myhostname = backup-server.local
mydestination = backup-server.local, localhost.localdomain, localhost
relayhost = [smtp.gmail.com]:587  # Replace with your SMTP server
Gmail SMTP authentication (if using Gmail)
smtp_sasl_auth_enable = yes
smtp_sasl_password_maps = hash:/etc/postfix/sasl_passwd
smtp_sasl_security_options = noanonymous
smtp_tls_security_level = encrypt
smtp_tls_CAfile = /etc/ssl/certs/ca-certificates.crt
```

Set up authentication (if using Gmail):

```
sudo nano /etc/postfix/sasl_passwd
```

Add:

```
[smtp.gmail.com]:587 youremail@gmail.com:yourpassword
```

Generate the password database and set permissions:

```
sudo postmap /etc/postfix/sasl_passwd
sudo chmod 600 /etc/postfix/sasl_passwd
sudo chmod 600 /etc/postfix/sasl_passwd.db
```

Restart Postfix:

```
sudo systemctl restart postfix
```

Note: For Gmail, you'll need to enable “Less secure app access” or create an App Password if you have 2FA enabled.

### 2.7 Configure Prometheus

Edit the Prometheus configuration file:

```
sudo nano /etc/prometheus/prometheus.yml
```

Add this configuration:

```
global:
  scrape_interval: 15s
scrape_configs:

job_name: 'prometheus'
static_configs:

targets: ['localhost:9090']


job_name: 'node_exporter'
static_configs:

targets: ['localhost:9100', 'client-system:9100']
```

Restart Prometheus:

```
sudo systemctl restart prometheus
```

### 2.8 Configure Grafana

Start and enable Grafana:

```
sudo systemctl enable grafana-server
sudo systemctl start grafana-server
```

Access Grafana at http://backup-server:3000 (default credentials: admin/admin)

### 2.9 Configure Nginx for Web Interface

Create a new Nginx configuration file:

```
sudo nano /etc/nginx/sites-available/backup-interface
```

Add this configuration:

```
server {
    listen 80;
    server_name backup-server;
location / {
    proxy_pass http://localhost:5000;
    proxy_set_header Host $host;
    proxy_set_header X-Real-IP $remote_addr;
}
}
```

Enable the site:

```
sudo ln -s /etc/nginx/sites-available/backup-interface /etc/nginx/sites-
enabled/
sudo rm /etc/nginx/sites-enabled/default  # Remove default site
sudo nginx -t  # Test configuration
sudo systemctl restart nginx
```

## 3. Client System Configuration

### 3.1 Install Required Packages

```
sudo apt install -y borgbackup python3-pip python3-dev libssl-dev libffi-dev
prometheus-node-exporter rclone
```

### 3.2 Generate SSH Key for Authentication

```
ssh-keygen -t ed25519 -C "client-backup"
```

Press Enter to accept the default file location and leave the passphrase empty for automated backups.

### 3.3 Copy SSH Key to Backup Server

```
ssh-copy-id backup-admin@backup-server
```

### 3.4 Install Borgmatic

```
sudo pip3 install borgmatic
```

### 3.5 Create Borgmatic Configuration

```
sudo mkdir -p /etc/borgmatic.d
sudo nano /etc/borgmatic.d/config.yaml
```

Add this configuration:

```
location:
    source_directories:
        - /home/business-user/documents
        - /home/business-user/important_files
        - /etc
repositories:
    - backup-admin@backup-server:/backup/repositories/client-system

exclude_patterns:
    - '*.temp'
    - '*.log'
    - '/home/*/Downloads'
    - '*.mp3'
    - '*.mp4'
storage:
compression: lz4
encryption_passphrase: "Admin"
archive_name_format: '{hostname}-{now:%Y-%m-%d-%H%M%S}'
retention:
keep_daily: 7
keep_weekly: 4
keep_monthly: 6
consistency:
checks:
- repository
- archives
check_last: 3
hooks:
before_backup:
- echo "Starting backup at $(date)"
after_backup:
- echo "Backup finished at $(date)"
on_error:
- echo "Error during backup at $(date)" | mail -s "Backup error for
${HOSTNAME}" business-user@localhost
```

### 3.6 Create Directories for Backup

```
mkdir -p /home/business-user/documents /home/business-user/important_files
chmod 755 /home/business-user/documents /home/business-user/important_files
```

### 3.7 Initialize Borg Repository

```
export BORG_PASSPHRASE="Admin"
borgmatic init --encryption repokey
```

### 3.8 Setup Borgmatic Cron Job for Automated Backups

```
sudo crontab -e
```

Add this line to run the backup daily at 2 AM:

```
0 2 * * * /usr/local/bin/borgmatic --verbosity 1
```

### 3.9 Install Vorta for GUI Management (Optional)

```
sudo apt install -y python3-setuptools python3-pyqt5
sudo pip3 install vorta
```

### 3.10 Configure Rclone for Cloud Backup

```
rclone config
```

Follow the interactive setup to configure your preferred cloud provider (Google Drive, Dropbox, etc.). After configuration, create a script for cloud sync:

```
sudo nano /usr/local/bin/cloud-sync.sh
```

Add:

```
#!/bin/bash
rclone sync /home/business-user/documents remote:backup/documents
rclone sync /home/business-user/important_files
remote:backup/important_files
```

Make it executable:

```
sudo chmod +x /usr/local/bin/cloud-sync.sh
```

Add to crontab to run daily:

```
sudo crontab -e
```

Add:

```
0 4 * * * /usr/local/bin/cloud-sync.sh
```

## 4. Web Interface Setup

### 4.1 Create a Flask Web Interface on Backup Server

```
sudo mkdir -p /opt/backup-interface
cd /opt/backup-interface
```

Create a Python virtual environment:

```
sudo apt install -y python3-venv
python3 -m venv venv
source venv/bin/activate
```

Install required packages:

```
pip install flask flask-login werkzeug
```

Create application directories:

```
mkdir -p templates static
```

Create the main application file:

```
sudo nano app.py
```

Add this Flask application code:

```
from flask import Flask, render_template, request, redirect, url_for, flash,
session
import os
import subprocess
import json
import datetime
from werkzeug.security import check_password_hash, generate_password_hash
app = Flask(name)
app.secret_key = os.urandom(24)
app.config['PERMANENT_SESSION_LIFETIME'] = datetime.timedelta(minutes=30)
Hard-coded user for simplicity (in production, use a database)
USERS = {
'admin': {
'password': generate_password_hash('Admin')
}
}
def get_backup_status():
try:
result = subprocess.run(['ssh', 'backup-admin@backup-server', 'borg',
'list', '/backup/repositories/client-system'],
capture_output=True, text=True)
if result.returncode == 0:
archives = result.stdout.strip().split('\n')
return {'status': 'success', 'archives': archives}
else:
return {'status': 'error', 'message': result.stderr}
except Exception as e:
return {'status': 'error', 'message': str(e)}
def get_system_health():
try:
# Check disk space
disk = subprocess.run(['ssh', 'backup-admin@backup-server', 'df', '-h',
'/backup'],
capture_output=True, text=True)
    # Check server load
    load = subprocess.run(['ssh', 'backup-admin@backup-server', 'uptime'],
                         capture_output=True, text=True)

    return {
        'status': 'success',
        'disk': disk.stdout,
        'load': load.stdout
    }
except Exception as e:
    return {'status': 'error', 'message': str(e)}
@app.route('/')
def index():
if 'username' not in session:
return redirect(url_for('login'))
return render_template('dashboard.html')
@app.route('/login', methods=['GET', 'POST'])
def login():
if request.method == 'POST':
username = request.form['username']
password = request.form['password']
    if username in USERS and
check_password_hash(USERS[username]['password'], password):
        session['username'] = username
        return redirect(url_for('index'))

    flash('Invalid username or password')

return render_template('login.html')
@app.route('/logout')
def logout():
session.pop('username', None)
return redirect(url_for('login'))
@app.route('/backup_status')
def backup_status():
if 'username' not in session:
return redirect(url_for('login'))
status = get_backup_status()
return render_template('backup_status.html', status=status)
@app.route('/health')
def health():
if 'username' not in session:
return redirect(url_for('login'))
health_data = get_system_health()
return render_template('health.html', health=health_data)
@app.route('/restore', methods=['GET', 'POST'])
def restore():
if 'username' not in session:
return redirect(url_for('login'))
```

```
if request.method == 'POST':
    archive = request.form['archive']
    path = request.form['path']
    destination = request.form['destination']

    try:
        cmd = f"ssh backup-admin@backup-server 'cd {destination} &&
BORG_PASSPHRASE=Admin borg extract /backup/repositories/client-
system::{archive} {path}'"
        result = subprocess.run(cmd, shell=True, capture_output=True,
text=True)

        if result.returncode == 0:
            flash('Files restored successfully!')
        else:
            flash(f'Error: {result.stderr}')

    except Exception as e:
        flash(f'Error: {str(e)}')

status = get_backup_status()
return render_template('restore.html', status=status)
if name == 'main':
app.run(host='0.0.0.0', port=5000)
```

Now create the necessary templates: Create login template:

```
sudo nano templates/login.html
```

Add:

```
<!DOCTYPE html>
<html>
<head>
    <title>Backup System Login</title>
    <style>
        body {
            font-family: Arial, sans-serif;
            background-color: #f4f4f4;
            margin: 0;
            padding: 0;
            display: flex;
            justify-content: center;
            align-items: center;
            height: 100vh;
        }
        .login-container {
            background: white;
            padding: 20px;
            border-radius: 5px;
            box-shadow: 0 0 10px rgba(0,0,0,0.1);
            width: 300px;
        }
        h2 {
            text-align: center;
            color: #333;
        }
        input[type="text"], input[type="password"], button {
            width: 100%;
            padding: 10px;
            margin: 8px 0;
            display: inline-block;
            border: 1px solid #ccc;
            border-radius: 4px;
            box-sizing: border-box;
        }
        button {
            background-color: #4CAF50;
            color: white;
            border: none;
            cursor: pointer;
        }
        button:hover {
            background-color: #45a049;
        }
        .flash-message {
            padding: 10px;
            margin: 10px 0;
            background-color: #f44336;
            color: white;
            border-radius: 4px;
        }
    </style>
</head>
<body>
    <div class="login-container">
        <h2>Backup System Login</h2>
    {% with messages = get_flashed_messages() %}
      {% if messages %}
        {% for message in messages %}
          <div class="flash-message">{{ message }}</div>
        {% endfor %}
      {% endif %}
    {% endwith %}

    <form method="POST">
        <div>
            <label for="username">Username:</label>
            <input type="text" id="username" name="username" required>
        </div>
        <div>
            <label for="password">Password:</label>
            <input type="password" id="password" name="password" required>
        </div>
        <button type="submit">Login</button>
    </form>
</div>
</body>
</html>
```

Create dashboard template:

```
sudo nano templates/dashboard.html
```

Add:

```
<!DOCTYPE html>
<html>
<head>
    <title>Backup System Dashboard</title>
    <style>
        body {
            font-family: Arial, sans-serif;
            margin: 0;
            padding: 0;
            background-color: #f4f4f4;
        }
        .container {
            width: 80%;
            margin: 0 auto;
            padding: 20px;
        }
        .header {
            background-color: #333;
            color: white;
            padding: 10px 20px;
            display: flex;
            justify-content: space-between;
            align-items: center;
        }
        .header h1 {
            margin: 0;
        }
        .nav {
            display: flex;
        }
        .nav a {
            color: white;
            margin-left: 15px;
            text-decoration: none;
        }
        .dashboard-grid {
            display: grid;
            grid-template-columns: 1fr 1fr;
            gap: 20px;
            margin-top: 20px;
        }
        .dashboard-item {
            background-color: white;
            padding: 20px;
            border-radius: 5px;
            box-shadow: 0 0 10px rgba(0,0,0,0.1);
        }
        .dashboard-item h2 {
            margin-top: 0;
            color: #333;
        }
        .btn {
            display: inline-block;
            background-color: #4CAF50;
            color: white;
            padding: 10px 15px;
            text-decoration: none;
            border-radius: 4px;
            margin-top: 10px;
        }
    </style>
</head>
<body>
    <div class="header">
        <h1>Backup System Dashboard</h1>
        <div class="nav">
            <a href="{{ url_for('backup_status') }}">Backup Status</a>
            <a href="{{ url_for('health') }}">System Health</a>
            <a href="{{ url_for('restore') }}">Restore Files</a>
            <a href="{{ url_for('logout') }}">Logout</a>
        </div>
    </div>
<div class="container">
    <div class="dashboard-grid">
        <div class="dashboard-item">
            <h2>Backup System Status</h2>
            <p>Welcome to the Secure Remote Backup System Dashboard. This
interface allows you to monitor and manage your backups.</p>
            <a href="{{ url_for('backup_status') }}" class="btn">View Backup
Status</a>
        </div>

        <div class="dashboard-item">
            <h2>System Health</h2>
            <p>Monitor disk space, server load, and other important metrics
of your backup system.</p>
            <a href="{{ url_for('health') }}" class="btn">Check System
Health</a>
```

```
        </div>

        <div class="dashboard-item">
            <h2>Restore Files</h2>
            <p>Restore files from your backups with a few simple steps.</p>
            <a href="{{ url_for('restore') }}" class="btn">Restore Files</a>
        </div>

        <div class="dashboard-item">
            <h2>Quick Links</h2>
            <ul>
                <li><a href="http://backup-server:3000"
target="_blank">Grafana Monitoring Dashboard</a></li>
                <li><a href="http://backup-server:9090"
target="_blank">Prometheus Metrics</a></li>
            </ul>
        </div>
    </div>
</div>
</body>
</html>
```

Create backup status template:

```
sudo nano templates/backup_status.html
```

Add:

```
<!DOCTYPE html>
<html>
<head>
    <title>Backup Status</title>
    <style>
        body {
            font-family: Arial, sans-serif;
            margin: 0;
            padding: 0;
            background-color: #f4f4f4;
        }
        .container {
            width: 80%;
            margin: 0 auto;
            padding: 20px;
        }
        .header {
            background-color: #333;
            color: white;
            padding: 10px 20px;
            display: flex;
            justify-content: space-between;
            align-items: center;
        }
        .header h1 {
            margin: 0;
        }
        .nav {
            display: flex;
        }
        .nav a {
            color: white;
            margin-left: 15px;
            text-decoration: none;
        }
        .content-box {
            background-color: white;
            padding: 20px;
            border-radius: 5px;
            box-shadow: 0 0 10px rgba(0,0,0,0.1);
            margin-top: 20px;
        }
        .content-box h2 {
            margin-top: 0;
            color: #333;
        }
        .form-group {
            margin-bottom: 15px;
        }
        .form-group label {
            display: block;
            margin-bottom: 5px;
        }
        .form-group input, .form-group select {
            width: 100%;
            padding: 8px;
            border: 1px solid #ddd;
            border-radius: 4px;
            box-sizing: border-box;
        }
        .btn {
            background-color: #4CAF50;
            color: white;
            padding: 10px 15px;
            border: none;
            border-radius: 4px;
            cursor: pointer;
        }
        .btn:hover {
            background-color: #45a049;
        }
        .flash-message {
            padding: 10px;
            margin: 10px 0;
            border-radius: 4px;
        }
        .flash-success {
            background-color: #dff0d8;
            color: #3c763d;
        }
        .flash-error {
            background-color: #f2dede;
            color: #a94442;
        }
    </style>
</head>
<body>
    <div class="header">
        <h1>Restore Files</h1>
        <div class="nav">
            <a href="{{ url_for('index') }}">Dashboard</a>
            <a href="{{ url_for('backup_status') }}">Backup Status</a>
            <a href="{{ url_for('health') }}">System Health</a>
            <a href="{{ url_for('logout') }}">Logout</a>
        </div>
    </div>
<div class="container">
    {% with messages = get_flashed_messages() %}
      {% if messages %}
        {% for message in messages %}
          <div class="flash-message {% if 'Error' in message %}flash-error{%
else %}flash-success{% endif %}">
            {{ message }}
          </div>
        {% endfor %}
      {% endif %}
    {% endwith %}

    <div class="content-box">
        <h2>Restore Files from Backup</h2>

        {% if status.status == 'success' and status.archives %}
            <form method="POST">
                <div class="form-group">
                    <label for="archive">Select Backup Archive:</label>
                    <select id="archive" name="archive" required>
                        {% for archive in status.archives %}
                            <option value="{{ archive }}">{{ archive
}}</option>
                        {% endfor %}
                    </select>
                </div>

                <div class="form-group">
                    <label for="path">File or Directory Path to Restore
```

```
(from backup):</label>
                    <input type="text" id="path" name="path"
placeholder="e.g., home/business-user/documents/file.txt" required>
                </div>

                <div class="form-group">
                    <label for="destination">Destination Path (where to
restore):</label>
                    <input type="text" id="destination" name="destination"
placeholder="e.g., /home/business-user/restored" required>
                </div>

                <button type="submit" class="btn">Restore Files</button>
            </form>
        {% else %}
            <p>No backup archives available for restoration.</p>
        {% endif %}
    </div>
</div>
</body>
</html>
```

### 4.2 Create a Systemd Service for the Web Interface

Create a systemd service file to ensure the web interface runs automatically at startup and restarts if it crashes:

```
sudo nano /etc/systemd/system/backup-interface.service
```

Add:

```
[Unit]
Description=Backup Web Interface
After=network.target
[Service]
User=backup-admin
WorkingDirectory=/opt/backup-interface
ExecStart=/opt/backup-interface/venv/bin/python /opt/backup-interface/app.py
Restart=always
[Install]
WantedBy=multi-user.target
```

Enable and start the service:

```
sudo systemctl enable backup-interface
sudo systemctl start backup-interface
```

# Testing the System

## 1. Creating Test Files on Client System

Create test files to verify the backup system works correctly:

```
cd /home/business-user/documents
echo "This is a test document" > test_doc1.txt
echo "This is another test document" > test_doc2.txt
cd /home/business-user/important_files
echo "This is an important file" > important1.txt
mkdir -p finance
echo "Financial data" > finance/data.txt
```

## 2. Running a Manual Backup Test

```
sudo borgmatic --verbosity 2
```

This should run without errors and create a backup on the server.

## 3. Verifying Backup Creation

On the backup server, check that the repository contains your backup:

```
export BORG_PASSPHRASE="Admin"
borg list /backup/repositories/client-system
```

You should see the newly created archive.

## 4. Testing the Web Interface

Access the web interface by visiting http://backup-server in your browser:

Login with username: admin, password: Admin Navigate to “Backup Status” to see your backups Navigate to “Restore Files” to test restoration

## 5. Testing File Restoration

### Through the Web Interface

Select a backup to restore Specify the path of a file to restore (e.g., home/business-user/documents/test_doc1.txt) Specify a destination folder (e.g., /home/business-user/restored) Click “Restore Files”

**Through the Command Line**

```
export BORG_PASSPHRASE="Admin"
# List archives
borg list backup-admin@backup-server:/backup/repositories/client-system
# Extract a specific file
borg extract backup-admin@backup-server:/backup/repositories/client-
system::ARCHIVE_NAME home/business-user/documents/test_doc1.txt
```

Replace ARCHIVE_NAME with the actual archive name from the list.

## 6. Testing Email Notifications

Modify the Borgmatic configuration to deliberately cause an error (e.g., change a path), then run the backup to verify that you receive an error notification email:

```
sudo nano /etc/borgmatic.d/config.yaml
```

Change one of the source directories to a non-existent path:

```
location:
    source_directories:
        - /home/business-user/nonexistent
```

Run borgmatic to test the error notification:

```
sudo borgmatic --verbosity 2
```

After verifying the email notification, don't forget to revert the configuration:

```
sudo nano /etc/borgmatic.d/config.yaml
```

## 7. Testing Automatic Backups

Wait for the cron job to run at 2 AM, or temporarily change the schedule to run sooner for testing:

```
sudo crontab -e
```

Change the cron entry to run a few minutes in the future:

```
# Run in 5 minutes from now
*/5 * * * * /usr/local/bin/borgmatic --verbosity 1
```

Don't forget to revert the cron entry afterward.

# Troubleshooting

## SSH Connection Issues

If you encounter SSH connection issues between the client and server:

Verify both systems can ping each other Check SSH service is running:

```
sudo systemctl status sshd
```

Ensure SSH keys are correctly copied:

```
ssh-copy-id backup-admin@backup-server
```

Verify permissions on ~/.ssh directory:

```
chmod 700 ~/.ssh && chmod 600 ~/.ssh/authorized_keys
```

## Backup Failures

If backups are failing:

Check borgmatic logs:

```
sudo borgmatic --verbosity 2
```

Verify the encryption passphrase is correctly set Ensure source directories exist and are readable Check disk space on the backup server:

```
df -h
```

## Web Interface Issues

If the web interface is not accessible:

Verify the service is running:

```
sudo systemctl status backup-interface
```

Check Nginx configuration:

```
sudo nginx -t
```

Look for errors in the logs:

```
sudo journalctl -u backup-interface
```

Ensure the correct ports are open:

```
sudo ufw status
```

## Email Notification Problems

If email notifications aren't working:

Check Postfix logs:

```
sudo journalctl -u postfix
```

Verify Postfix configuration:

```
sudo postconf -n
```

Test sending an email manually:

```
echo "Test email" | mail -s "Test" youremail@example.com
```

If using Gmail, ensure “Less secure app access” is enabled or use App Passwords

# Maintenance Procedures

## Regular Maintenance Tasks

### Daily

Check backup status through the web interface Verify email notifications are being sent Confirm client and server are communicating properly

### Weekly

Review system logs:

```
sudo journalctl -u backup-interface
```

Check disk space usage:

```
df -h
```

Verify all services are running:

```
sudo systemctl status nginx postfix prometheus grafana-server
```

### Monthly

Update all systems:

```
sudo apt update && sudo apt upgrade -y
```

Test backup restoration process Review retention policies Check for outdated dependencies

## Backup Repository Maintenance

Borg offers several maintenance commands for the backup repository:

### Checking Repository Consistency

```
export BORG_PASSPHRASE="Admin"
borg check backup-admin@backup-server:/backup/repositories/client-system
```

### Compacting Repository

This command optimizes repository space usage:

```
export BORG_PASSPHRASE="Admin"
borg compact backup-admin@backup-server:/backup/repositories/client-system
```

### Pruning Old Backups

While borgmatic handles this automatically, you can manually run:

```
sudo borgmatic prune --verbosity 2
```

## System Backup

Don't forget to back up the backup system configuration:

```
sudo tar -czf backup-system-config.tar.gz /etc/borgmatic.d
/etc/systemd/system/backup-interface.service /opt/backup-interface
```

# Security Considerations

## Encryption

backup system uses strong encryption to protect your data:

All backups are encrypted with AES-256 encryption Encryption is handled by Borg using the “repokey” mode The encryption passphrase should be stored securely and backed up separately Never store the encryption passphrase on the same system as the backups

## Network Security

To enhance network security:

Use SSH key-based authentication only (password authentication disabled) Consider restricting SSH access to specific IP addresses using a firewall Use strong, unique passwords for all user accounts Keep all systems updated with security patches

## Web Interface Security

The web interface has several security measures:

Authentication required to access all pages Password hashing using Werkzeug's secure functions Session timeout after 30 minutes of inactivity For production, add HTTPS using Let's Encrypt:

```
sudo certbot --nginx -d backup-server.yourdomain.com
```

## Firewall Configuration

To secure your systems further, configure a firewall:

```
# On both systems
sudo apt install -y ufw
On backup server
sudo ufw allow ssh
sudo ufw allow http
sudo ufw allow 3000  # For Grafana
sudo ufw allow 9100  # For Prometheus Node Exporter
On client system
sudo ufw allow ssh
sudo ufw allow 9100  # For Prometheus Node Exporter
Enable the firewall on both systems
sudo ufw enable
```

# Appendix

## Glossary

BorgBackup: Deduplicating backup program with compression and encryption Borgmatic: Command-line tool for automating BorgBackup tasks Deduplicate: Store duplicate data only once to save space Incremental Backup: Only back up changes since the last backup Repository: Storage location for backups Retention Policy: Rules defining how long backups are kept

## References

BorgBackup Documentation

Borgmatic Documentation

Rclone Documentation

Nginx Documentation