Proxmox LXC Auto-Enrollment Guide

Overview

PatchMon's Proxmox Auto-Enrollment feature enables you to automatically discover and enroll LXC containers from your Proxmox hosts into PatchMon for centralized patch management. This eliminates manual host registration and ensures comprehensive coverage of your Proxmox infrastructure.

What It Does

- **Automatically discovers** running LXC containers on Proxmox hosts
- **Bulk enrolls** containers into PatchMon without manual intervention
- **Installs agents** inside each container automatically
- **Assigns to host groups** based on token configuration
- **Tracks enrollment** with full audit logging

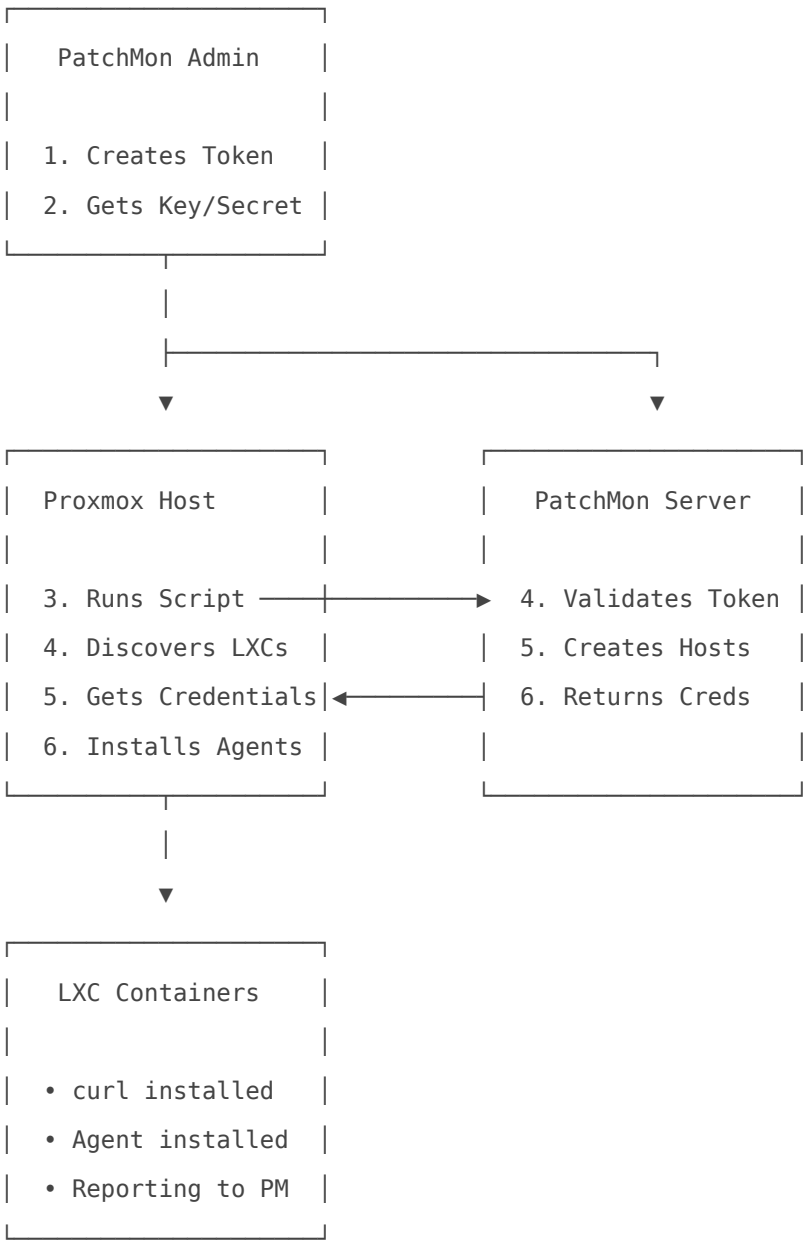
Key Benefits

- **Zero-Touch Enrollment** - Run once, enroll all containers
- **Secure by Design** - Token-based authentication with hashed secrets
- **Rate Limited** - Prevents abuse with per-day host limits
- **IP Restricted** - Optional IP whitelisting for enhanced security
- **Fully Auditable** - Tracks who enrolled what and when
- **Safe to Rerun** - Already-enrolled containers are automatically skipped

Table of Contents

How It Works

Architecture Overview



Enrollment Process (Step by Step)

1. **Admin creates auto-enrollment token** in PatchMon UI
   - Configures rate limits, IP restrictions, host group assignment
   - Receives `token_key` and `token_secret` (shown only once!)
2. **Admin runs enrollment script** on Proxmox host
   - Script authenticated with auto-enrollment token
   - Discovers all running LXC containers using `pct list`
3. **For each container**, the script:
   - Gathers hostname, IP address, OS information, machine ID
   - Calls PatchMon API to create host entry
   - Receives unique `api_id` and `api_key` for that container
   - Uses `pct exec` to enter the container
   - Installs curl if missing
   - Downloads and runs PatchMon agent installer
   - Agent authenticates with container-specific credentials
4. **Containers appear in PatchMon** with full patch tracking enabled

Two-Tier Security Model

1. Auto-Enrollment Token (Script → PatchMon)

- **Purpose**: Create new host entries
- **Scope**: Limited to enrollment operations only
- **Storage**: Secret is hashed in database
- **Lifespan**: Reusable until revoked/expired
- **Security**: Rate limits + IP restrictions

2. Host API Credentials (Agent → PatchMon)

- **Purpose**: Report patches, send data, receive commands
- **Scope**: Per-host unique credentials
- **Storage**: API key is hashed (bcrypt) in database
- **Lifespan**: Permanent for that host
- **Security**: Host-specific, can be regenerated

Why This Matters:

- Compromised enrollment token ≠ compromised hosts
- Compromised host credential ≠ compromised enrollment
- Revoked enrollment token = no new enrollments (existing hosts unaffected)
- Lost credentials = create new token, don't affect existing infrastructure

Prerequisites

PatchMon Server Requirements

- PatchMon version with auto-enrollment support
- Admin user with "Manage Settings" permission
- Network accessible from Proxmox hosts

Proxmox Host Requirements

- Proxmox VE installed and running
- One or more LXC containers (VMs not supported)
- Root access to Proxmox host
- Network connectivity to PatchMon server
- Required commands: `pct`, `curl`, `jq`, `bash`

Container Requirements

- Running state (stopped containers are skipped)
- Debian-based or RPM-based Linux distribution
- Network connectivity to PatchMon server
- Package manager (apt/yum/dnf) functional

Network Requirements

Source	Destination	Port	Protocol	Purpose
Proxmox Host	PatchMon Server	443 (HTTPS)	TCP	Enrollment API calls
LXC Containers	PatchMon Server	443 (HTTPS)	TCP	Agent installation & reporting

Firewall Notes:

- Outbound only connections (no inbound ports needed)
- HTTPS recommended (HTTP supported for internal networks)
- Self-signed certificates supported with `-k` flag

Quick Start

1. Create Token (In PatchMon UI)

1. Go to **Settings → Integrations → Auto-Enrollment & API** tab
2. Click **"New Token"**
3. Configure:
   - **Name**: "Production Proxmox"
   - **Max Hosts/Day**: 100
   - **Host Group**: Select target group
   - **IP Restriction**: Your Proxmox host IP
4. **Save credentials immediately** (shown only once!)

2. One-Line Enrollment (On Proxmox Host)

```
curl -s "https://patchmon.example.com/api/v1/auto-enrollment/script?type=proxmox-lxc&token_key=YOUR_KEY&token_secret=YOUR_SECRET" | bash
```

That's it! All running LXC containers will be enrolled and the PatchMon agent installed.

3. Verify in PatchMon

- Go to **Hosts** page
- See your containers listed with "pending" status
- Agent connects automatically after installation (usually within seconds)
- Status changes to "active" with package data

Step-by-Step Setup

Step 1: Create Auto-Enrollment Token

Via PatchMon Web UI

1. **Log in to PatchMon** as an administrator
2. **Navigate to Settings**

   ```
   Dashboard → Settings → Integrations → Auto-Enrollment & API tab
   ```

3. **Click "New Token"** button
4. **Fill in token details:**

Field	Value	Required	Description

Token Name	`Proxmox Production`	Yes	Descriptive name for this token
Max Hosts Per Day	`100`	Yes	Rate limit (1-1000)
Default Host Group	`Proxmox LXC`	No	Auto-assign enrolled hosts
Allowed IP Addresses	`192.168.1.10`	No	Comma-separated IPs
Expiration Date	`2027-01-01`	No	Auto-disable after date

5. **Click "Create Token"**
6. **CRITICAL: Save Credentials Now!**
   You'll see a success modal with:

```
Token Key:    patchmon_ae_a1b2c3d4e5f6...
Token Secret: 8f7e6d5c4b3a2f1e0d9c8b7a...
```

   Copy both values immediately! They cannot be retrieved later.
   Pro Tip: Copy the one-line installation command shown in the modal - it has credentials pre-filled.

Step 2: Prepare Proxmox Host

Install Required Dependencies

```
# SSH to your Proxmox host
ssh root@proxmox-host

# Install jq (JSON processor)
apt-get update && apt-get install -y jq curl

# Verify installations
which pct jq curl
# Should show paths for all three commands
```

Download Enrollment Script

Method A: Direct Download from PatchMon (Recommended)

```
# Download with credentials embedded (copy from PatchMon UI)
curl -s "https://patchmon.example.com/api/v1/auto-enrollment/script?type=proxmox-
lxc&token_key=YOUR_KEY&token_secret=YOUR_SECRET" \
    -o /root/proxmox_auto_enroll.sh
```

```
chmod +x /root/proxmox_auto_enroll.sh
```

Method B: Manual Configuration

```
# Download script template
cd /root
wget https://raw.githubusercontent.com/PatchMon/PatchMon/main/agents/proxmox_auto_enroll.sh
chmod +x proxmox_auto_enroll.sh

# Edit configuration
nano proxmox_auto_enroll.sh

# Update these lines:
PATCHMON_URL="https://patchmon.example.com"
AUTO_ENROLLMENT_KEY="patchmon_ae_your_key_here"
AUTO_ENROLLMENT_SECRET="your_secret_here"
```

Step 3: Test with Dry Run

Always test first!

```
# Dry run shows what would happen without making changes
DRY_RUN=true ./proxmox_auto_enroll.sh
```

Expected output:

```
[INFO] Found 5 LXC container(s)
[INFO] Processing LXC 100: webserver (status: running)
[INFO]   [DRY RUN] Would enroll: proxmox-webserver
[INFO] Processing LXC 101: database (status: running)
[INFO]   [DRY RUN] Would enroll: proxmox-database
...
[INFO] Successfully Enrolled:  5 (dry run)
```

Step 4: Run Actual Enrollment

```
# Enroll all containers
./proxmox_auto_enroll.sh
```

Monitor the output:

- Green `[SUCCESS]` = Container enrolled and agent installed
- Yellow `[WARN]` = Container skipped (already enrolled or stopped)
- Red `[ERROR]` = Failure (check troubleshooting section)

Step 5: Verify in PatchMon

1. **Go to Hosts page** in PatchMon UI
2. **Look for newly enrolled containers** (names prefixed with "proxmox-")
3. **Initial status is "pending"** (normal!)
4. **Agent connects automatically** after installation (usually within seconds)
5. **Status changes to "active"** with package data populated

Troubleshooting: If status stays "pending" after a couple of minutes, see Agent Not Reporting section.

Usage Examples

Basic Enrollment

```
# Enroll all running LXC containers
./proxmox_auto_enroll.sh
```

Dry Run Mode

```
# Preview what would be enrolled (no changes made)
DRY_RUN=true ./proxmox_auto_enroll.sh
```

Debug Mode

```
# Show detailed logging for troubleshooting
DEBUG=true ./proxmox_auto_enroll.sh
```

Custom Host Prefix

```
# Prefix container names (e.g., "prod-webserver" instead of "webserver")
HOST_PREFIX="prod-" ./proxmox_auto_enroll.sh
```

Include Stopped Containers

```
# Also process stopped containers (enrollment only, agent install fails)
SKIP_STOPPED=false ./proxmox_auto_enroll.sh
```

Force Install Mode (Broken Packages)

If containers have broken packages (CloudPanel, WHM, cPanel, etc.) that block `apt-get`:

```
# Bypass broken packages during agent installation
FORCE_INSTALL=true ./proxmox_auto_enroll.sh
```

Or use the force parameter when downloading:

```
curl -s "https://patchmon.example.com/api/v1/auto-enrollment/script?type=proxmox-
lxc&token_key=KEY&token_secret=SECRET&force=true" | bash
```

What force mode does:

- Skips `apt-get update` if broken packages detected
- Only installs missing critical tools (jq, curl, bc)
- Uses `--fix-broken --yes` flags safely
- Validates installations before proceeding

Scheduled Enrollment (Cron)

Automatically enroll new containers on a schedule. Since cron runs with a minimal environment (limited `PATH`, no user variables), you need to ensure the crontab has the correct environment set up for the script to find required commands like `pct`, `curl`, and `jq`.

Setting Up the Crontab

Edit the root crontab:

```
crontab -e
```

Add the following. The `PATH` and environment variables at the top are essential - without them the script will fail because cron does not inherit your shell's environment:

```
# === PatchMon Auto-Enrollment Environment ===
# Cron uses a minimal PATH by default (/usr/bin:/bin). The enrollment script
# requires pct, curl, and jq which may live in /usr/sbin or other paths.
```

```
# Set a full PATH so all commands are found.
SHELL=/bin/bash
PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin

# Enrollment credentials (required by the script)
PATCHMON_URL=https://patchmon.example.com
AUTO_ENROLLMENT_KEY=patchmon_ae_your_key_here
AUTO_ENROLLMENT_SECRET=your_secret_here

# Optional overrides
# HOST_PREFIX=proxmox-
# FORCE_INSTALL=false
# CURL_FLAGS=-sk

# === Schedule ===
# Run daily at 2 AM
0 2 * * * /root/proxmox_auto_enroll.sh >> /var/log/patchmon-enroll.log 2>&1

# Or hourly for dynamic environments where containers are created frequently
# 0 * * * * /root/proxmox_auto_enroll.sh >> /var/log/patchmon-enroll.log 2>&1
```

Why This Matters

Cron does not load your interactive shell profile (`~/.bashrc`, `~/.profile`, etc.). This means:

What cron is missing	Impact	Fix
`PATH` only includes `/usr/bin:/bin`	`pct` not found (lives in `/usr/sbin`)	Set `PATH` at top of crontab
No exported variables	`PATCHMON_URL`, credentials are empty	Define them in crontab or use a wrapper
No TTY	Colour output codes may cause log clutter	Redirect to log file with `2>&1`

Alternative: Wrapper Script

If you prefer not to put credentials in the crontab, create a wrapper script instead:

```
cat > /root/patchmon_enroll_cron.sh << 'EOF'
#!/bin/bash
# Wrapper that sets the environment for cron execution

export PATH="/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin"
```

```
export PATCHMON_URL="https://patchmon.example.com"
export AUTO_ENROLLMENT_KEY="patchmon_ae_your_key_here"
export AUTO_ENROLLMENT_SECRET="your_secret_here"
# export HOST_PREFIX="proxmox-"
# export CURL_FLAGS="-sk"

/root/proxmox_auto_enroll.sh
EOF

chmod 700 /root/patchmon_enroll_cron.sh
```

Then reference the wrapper in crontab:

```
0 2 * * * /root/patchmon_enroll_cron.sh >> /var/log/patchmon-enroll.log 2>&1
```

Make sure the wrapper script is only readable by root (`chmod 700`) since it contains secrets.

Log Rotation

For long-running cron schedules, consider adding log rotation to prevent unbounded log growth:

```
cat > /etc/logrotate.d/patchmon-enroll << 'EOF'
/var/log/patchmon-enroll.log {
    weekly
    rotate 4
    compress
    missingok
    notifempty
}
EOF
```

Verifying Cron is Working

```
# Check the cron job is registered
crontab -l | grep patchmon

# Check recent cron execution logs
grep patchmon /var/log/syslog | tail -n 20

# Check enrollment log output
tail -f /var/log/patchmon-enroll.log
```

Already-enrolled containers are automatically skipped on each run, so there is no risk of duplicates or errors from repeated execution.

Multi-Environment Setup

```
# Production environment (uses prod token)
export PATCHMON_URL="https://patchmon.example.com"
export AUTO_ENROLLMENT_KEY="patchmon_ae_prod_..."
export AUTO_ENROLLMENT_SECRET="prod_secret..."
export HOST_PREFIX="prod-"
./proxmox_auto_enroll.sh

# Development environment (uses dev token with different host group)
export AUTO_ENROLLMENT_KEY="patchmon_ae_dev_..."
export AUTO_ENROLLMENT_SECRET="dev_secret..."
export HOST_PREFIX="dev-"
./proxmox_auto_enroll.sh
```

Configuration Options

Environment Variables

All configuration can be set via environment variables:

Variable	Default	Description	Example
`PATCHMON_URL`	Required	PatchMon server URL	`https://patchmon.example.com`
`AUTO_ENROLLMENT_KEY`	Required	Token key from PatchMon	`patchmon_ae_abc123...`
`AUTO_ENROLLMENT_SECRET`	Required	Token secret from PatchMon	`def456ghi789...`
`CURL_FLAGS`	`-s`	Curl options	`-sk` (for self-signed SSL)
`DRY_RUN`	`false`	Preview mode (no changes)	`true` / `false`
`HOST_PREFIX`	`""`	Prefix for host names	`proxmox-`, `prod-`, etc.
`SKIP_STOPPED`	`true`	Skip stopped containers	`true` / `false`
`FORCE_INSTALL`	`false`	Bypass broken packages	`true` / `false`
`DEBUG`	`false`	Enable debug logging	`true` / `false`

Script Configuration Section

Or edit the script directly:

```
# ===== CONFIGURATION =====
PATCHMON_URL="${PATCHMON_URL:-https://patchmon.example.com}"
AUTO_ENROLLMENT_KEY="${AUTO_ENROLLMENT_KEY:-your_key_here}"
AUTO_ENROLLMENT_SECRET="${AUTO_ENROLLMENT_SECRET:-your_secret_here}"
CURL_FLAGS="${CURL_FLAGS:--s}"
DRY_RUN="${DRY_RUN:-false}"
HOST_PREFIX="${HOST_PREFIX:-}"
SKIP_STOPPED="${SKIP_STOPPED:-true}"
FORCE_INSTALL="${FORCE_INSTALL:-false}"
```

Token Configuration (PatchMon UI)

Configure tokens in **Settings → Integrations → Auto-Enrollment & API**:

General Settings:

- **Token Name**: Descriptive identifier
- **Active Status**: Enable/disable without deleting
- **Expiration Date**: Auto-disable after date

Security Settings:

- **Max Hosts Per Day**: Rate limit (resets daily at midnight)
- **Allowed IP Addresses**: Comma-separated IP whitelist
- **Default Host Group**: Auto-assign enrolled hosts

Usage Statistics:

- **Hosts Created Today**: Current daily count
- **Last Used**: Timestamp of most recent enrollment
- **Created By**: Admin user who created token
- **Created At**: Token creation timestamp

Security Best Practices

Token Management

1. **Store Securely**
   - Save credentials in password manager (1Password, LastPass, etc.)
   - Never commit to version control
   - Use environment variables or secure config management (Vault)
2. **Principle of Least Privilege**
   - Create separate tokens for prod/dev/staging
   - Use different tokens for different Proxmox clusters
   - Set appropriate rate limits per environment
3. **Regular Rotation**
   - Rotate tokens every 90 days
   - Disable unused tokens immediately
   - Monitor token usage for anomalies
4. **IP Restrictions**
   - Always set `allowed_ip_ranges` in production
   - Update if Proxmox host IPs change
   - Use VPN/private network IPs when possible
5. **Expiration Dates**
   - Set expiration for temporary/testing tokens
   - Review and extend before expiration
   - Delete expired tokens to reduce attack surface

Network Security

1. **Use HTTPS**
   - Always use encrypted connections in production
   - Use valid SSL certificates (avoid `-k` flag)
   - Self-signed OK for internal/testing environments
2. **Network Segmentation**
   - Run enrollment over private network if possible
   - Use proper firewall rules
   - Restrict PatchMon server access to known IPs

Access Control

1. **Admin Permissions**
   - Only admins with "Manage Settings" can create tokens
   - Regular users cannot see token secrets
   - Use role-based access control (RBAC)
2. **Audit Logging**
   - Monitor token creation/deletion in PatchMon logs
   - Track enrollment activity per token
   - Review host notes for enrollment source
3. **Container Security**
   - Ensure containers have minimal privileges

- Don't run enrollment as unprivileged user
- Use unprivileged containers where possible (enrollment still works)

Incident Response

If a token is compromised:

1. **Immediately disable** the token in PatchMon UI
   - Settings → Integrations → Auto-Enrollment & API → Toggle "Disable"
2. **Review recently enrolled hosts**
   - Check host notes for token name and enrollment date
   - Verify all recent enrollments are legitimate
   - Delete any suspicious hosts
3. **Create new token**
   - Generate new credentials
   - Update Proxmox script with new credentials
   - Test enrollment with dry run
4. **Investigate root cause**
   - How were credentials exposed?
   - Update procedures to prevent recurrence
   - Consider additional security measures
5. **Delete old token**
   - After verifying new token works
   - Document incident in change log

Troubleshooting

Common Errors and Solutions

Error: "pct command not found"

Symptom:

```
[ERROR] This script must run on a Proxmox host (pct command not found)
```

Cause: Script is running on a non-Proxmox machine

Solution:

```
# SSH to Proxmox host first
ssh root@proxmox-host
cd /root
```

```
./proxmox_auto_enroll.sh
```

Error: "Auto-enrollment credentials required"

Symptom:

```
[ERROR] Failed to enroll hostname - HTTP 401
Response: {"error":"Auto-enrollment credentials required"}
```

Cause: The `X-Auto-Enrollment-Key` and/or `X-Auto-Enrollment-Secret` headers are missing from the request

Solution:

1. Verify the script has `AUTO_ENROLLMENT_KEY` and `AUTO_ENROLLMENT_SECRET` set
2. Check for extra spaces/newlines in credentials
3. Ensure token_key starts with `patchmon_ae_`
4. Regenerate token if credentials lost

```
# Test credentials manually
curl -X POST \
  -H "X-Auto-Enrollment-Key: YOUR_KEY" \
  -H "X-Auto-Enrollment-Secret: YOUR_SECRET" \
  -H "Content-Type: application/json" \
  -d '{"friendly_name":"test","machine_id":"test"}' \
  https://patchmon.example.com/api/v1/auto-enrollment/enroll
```

Error: "Invalid or inactive token" / "Invalid token secret"

Symptom:

```
[ERROR] Failed to enroll hostname - HTTP 401
Response: {"error":"Invalid or inactive token"}
```

or

```
[ERROR] Failed to enroll hostname - HTTP 401
Response: {"error":"Invalid token secret"}
```

Cause: Token key not found or disabled (`Invalid or inactive token`), or secret doesn't match (`Invalid token secret`), or token has expired (`Token expired`)

Solution:

1. Check token status in PatchMon UI (Settings → Integrations)
2. Enable if disabled
3. Extend expiration if expired
4. Verify the secret matches the one shown when the token was created
5. Create new token if credentials are lost (secrets cannot be retrieved)

Error: "Rate limit exceeded"

Symptom:

```
[ERROR] Rate limit exceeded - maximum hosts per day reached
```

Cause: Token's `max_hosts_per_day` limit reached

Solution:

```
# Option 1: Wait until tomorrow (limit resets at midnight)
date
# Check current time, wait until 00:00

# Option 2: Increase limit in PatchMon UI
# Settings → Integrations → Edit Token → Max Hosts Per Day: 200

# Option 3: Create additional token for large enrollments
```

Error: "IP address not authorized"

Symptom:

```
[ERROR] Failed to enroll hostname - HTTP 403
Response: {"error":"IP address not authorized for this token"}
```

Cause: Proxmox host IP not in token's `allowed_ip_ranges`

Solution:

1. Find your Proxmox host IP:
   ```
   ip addr show | grep 'inet ' | grep -v 127.0.0.1
   ```
2. Update token in PatchMon UI:
   - Settings → Integrations → Edit Token
   - Allowed IP Addresses: Add your IP
3. Or remove IP restriction entirely (not recommended for production)

Error: "jq: command not found"

Symptom:

```
[ERROR] Required command 'jq' not found. Please install it first.
```

Cause: Missing dependency

Solution:

```
# Debian/Ubuntu
apt-get update && apt-get install -y jq

# CentOS/RHEL
yum install -y jq

# Alpine
apk add --no-cache jq
```

Error: "Failed to install agent in container"

Symptom:

```
[WARN] Failed to install agent in container-name (exit: 1)
Install output: E: Unable to locate package curl
```

Cause: Agent installation failed inside LXC container

Solutions:

A. Network connectivity issue:

```
# Test from Proxmox host
pct exec 100 -- ping -c 3 patchmon.example.com

# Test from inside container
pct enter 100
curl -I https://patchmon.example.com
exit
```

B. Package manager issue:

```
# Enter container
pct enter 100

# Update package lists
apt-get update
# or
yum makecache

# Try manual agent install
curl https://patchmon.example.com/api/v1/hosts/install \
  -H "X-API-ID: patchmon_xxx" \
  -H "X-API-KEY: xxx" | bash
```

C. Unsupported OS:

- Agent supports: Ubuntu, Debian, CentOS, RHEL, Rocky Linux, AlmaLinux, Alpine
- Check `/etc/os-release` in container
- Manually install on other distributions

D. Broken packages (use force mode):

```
FORCE_INSTALL=true ./proxmox_auto_enroll.sh
```

Error: SSL Certificate Problems

Symptom:

```
curl: (60) SSL certificate problem: self signed certificate
```

Cause: Self-signed certificate on PatchMon server

Solution:

```
# Use -k flag to skip certificate verification
export CURL_FLAGS="-sk"
./proxmox_auto_enroll.sh
```

Better solution: Install valid SSL certificate on PatchMon server using Let's Encrypt or corporate CA

Warning: Container Already Enrolled

Symptom:

```
[INFO] ✓ Host already enrolled and agent ping successful - skipping enrollment
```

Cause: The script detected an existing agent configuration (`/etc/patchmon/config.yml` and `/etc/patchmon/credentials.yml`) inside the container and the agent successfully pinged the PatchMon server.

This is normal! The script safely skips already-enrolled hosts. No action needed.

If you need to re-enroll:

1. Delete host from PatchMon UI (Hosts page)
2. Remove agent config inside the container: `pct exec <vmid> -- rm -rf /etc/patchmon/`
3. Rerun enrollment script

Agent Not Reporting

If containers show "pending" status after enrollment:

1. Check agent service is running:

```
pct enter 100

# For systemd-based containers
systemctl status patchmon-agent.service

# For OpenRC-based containers (Alpine)
rc-service patchmon-agent status

# For containers without init systems (crontab fallback)
ps aux | grep patchmon-agent
```

2. Check agent files exist:

```
ls -la /etc/patchmon/
# Should show: config.yml and credentials.yml

ls -la /usr/local/bin/patchmon-agent
# Should show the agent binary
```

3. Check agent logs:

```
# Systemd journal logs
journalctl -u patchmon-agent.service --no-pager -n 50

# Or check the agent log file
cat /etc/patchmon/logs/patchmon-agent.log
```

4. Test agent connectivity:

```
/usr/local/bin/patchmon-agent ping
# Should show success if credentials and connectivity are valid
```

5. Verify credentials:

```
cat /etc/patchmon/credentials.yml
# Should show api_id and api_key

cat /etc/patchmon/config.yml
# Should show patchmon_server URL
```

6. Restart the agent service:

```
# Systemd
systemctl restart patchmon-agent.service

# OpenRC
rc-service patchmon-agent restart
```

Debug Mode

Enable detailed logging:

```
DEBUG=true ./proxmox_auto_enroll.sh
```

Debug output includes:

- API request/response bodies
- Container command execution details
- Detailed error messages
- curl verbose output

Getting Help

If issues persist:

1. **Check PatchMon server logs:**
   ```
   tail -f /path/to/patchmon/backend/logs/error.log
   ```
2. **Create GitHub issue** with:
   - PatchMon version
   - Proxmox version
   - Script output (redact credentials!)
   - Debug mode output
   - Server logs (if accessible)
3. **Join Discord community** for real-time support

Advanced Usage

Selective Enrollment

Enroll only specific containers:

```
# Only enroll containers 100-199
nano proxmox_auto_enroll.sh

# Add after line "while IFS= read -r line; do"
vmid=$(echo "$line" | awk '{print $1}')
if [[ $vmid -lt 100 ]] || [[ $vmid -gt 199 ]]; then
    continue
fi
```

Or use container name filtering:

```
# Only enroll containers with "prod" in name
if [[ ! "$name" =~ prod ]]; then
    continue
fi
```

Custom Host Naming

Advanced naming strategies:

```
# Include Proxmox node name
HOST_PREFIX="$(hostname)-"
# Result: proxmox01-webserver, proxmox02-database

# Include datacenter/location
HOST_PREFIX="dc1-"
# Result: dc1-webserver, dc1-database

# Include environment and node
HOST_PREFIX="prod-$(hostname | cut -d. -f1)-"
# Result: prod-px01-webserver
```

Multi-Node Proxmox Cluster

For Proxmox clusters with multiple nodes:

Option 1: Same token, different prefix per node

```
# On node 1
HOST_PREFIX="node1-" ./proxmox_auto_enroll.sh

# On node 2
HOST_PREFIX="node2-" ./proxmox_auto_enroll.sh
```

Option 2: Different tokens per node

- Create token for each node with different default host groups
- Node 1 → "Proxmox Node 1" group
- Node 2 → "Proxmox Node 2" group

Option 3: Centralized automation

```
#!/bin/bash
# central_enroll.sh

NODES=(
  "root@proxmox01.example.com"
  "root@proxmox02.example.com"
  "root@proxmox03.example.com"
)
```

```
for node in "${NODES[@]}"; do
  echo "Enrolling containers from $node..."
  ssh "$node" "bash /root/proxmox_auto_enroll.sh"
done
```

Integration with Infrastructure as Code

Ansible Playbook:

```
---
- name: Enroll Proxmox LXC containers in PatchMon
  hosts: proxmox_hosts
  become: yes
  tasks:
    - name: Install dependencies
      apt:
        name:
          - curl
          - jq
        state: present

    - name: Download enrollment script
      get_url:
        url: "{{ patchmon_url }}/api/v1/auto-enrollment/script?type=proxmox-lxc&token_key={{
token_key }}&token_secret={{ token_secret }}"
        dest: /root/proxmox_auto_enroll.sh
        mode: '0700'

    - name: Run enrollment
      command: /root/proxmox_auto_enroll.sh
      register: enrollment_output

    - name: Show enrollment results
      debug:
        var: enrollment_output.stdout_lines
```

Terraform (with null_resource):

```
resource "null_resource" "patchmon_enrollment" {
  triggers = {
    cluster_instance_ids = join(",", proxmox_lxc.containers.*.vmid)
  }

  provisioner "remote-exec" {
    connection {
      host = var.proxmox_host
      user = "root"
      private_key = file(var.ssh_key_path)
    }

    inline = [
      "apt-get install -y jq",
      "curl -s '${var.patchmon_url}/api/v1/auto-enrollment/script?type=proxmox-
lxc&token_key=${var.token_key}&token_secret=${var.token_secret}' | bash"
    ]
  }
}
```

Bulk API Enrollment

For very large deployments (100+ containers), use the bulk API endpoint directly:

```
#!/bin/bash
# bulk_enroll.sh

# Gather all container info
containers_json=$(pct list | tail -n +2 | while read -r line; do
  vmid=$(echo "$line" | awk '{print $1}')
  name=$(echo "$line" | awk '{print $3}')

  echo "{\"friendly_name\":\"$name\",\"machine_id\":\"proxmox-lxc-$vmid\"}"
done | jq -s '.')

# Send bulk enrollment request
curl -X POST \
  -H "X-Auto-Enrollment-Key: $AUTO_ENROLLMENT_KEY" \
  -H "X-Auto-Enrollment-Secret: $AUTO_ENROLLMENT_SECRET" \
```

```
  -H "Content-Type: application/json" \
  -d "{\"hosts\":$containers_json}" \
  "$PATCHMON_URL/api/v1/auto-enrollment/enroll/bulk"
```

Benefits:

- Single API call for all containers
- Faster for 50+ containers
- Partial success supported (individual failures don't block others)

Limitations:

- Max 50 hosts per request
- Does not install agents (must be done separately)
- Less detailed error reporting per host

Webhook-Triggered Enrollment

Trigger enrollment from PatchMon webhook (requires custom setup):

```
#!/bin/bash
# webhook_listener.sh

# Simple webhook listener
while true; do
  # Listen for webhook on port 9000
  nc -l -p 9000 -c 'echo -e "HTTP/1.1 200 OK\n\n"; /root/proxmox_auto_enroll.sh'
done
```

Then configure PatchMon (or monitoring system) to call webhook when conditions are met.

API Reference

Admin Endpoints (Authentication Required)

All admin endpoints require JWT authentication:

```
Authorization: Bearer <jwt_token>
```

Create Token

Endpoint: `POST /api/v1/auto-enrollment/tokens`

Request:

```
{
  "token_name": "Proxmox Production",
  "max_hosts_per_day": 100,
  "default_host_group_id": "uuid",
  "allowed_ip_ranges": ["192.168.1.10", "10.0.0.5"],
  "expires_at": "2026-12-31T23:59:59Z",
  "metadata": {
    "integration_type": "proxmox-lxc",
    "environment": "production"
  }
}
```

Response: `201 Created`

```
{
  "message": "Auto-enrollment token created successfully",
  "token": {
    "id": "uuid",
    "token_name": "Proxmox Production",
    "token_key": "patchmon_ae_abc123...",
    "token_secret": "def456...",  // Only shown here!
    "max_hosts_per_day": 100,
    "default_host_group": {
      "id": "uuid",
      "name": "Proxmox LXC",
      "color": "#3B82F6"
    },
    "created_by": {
      "id": "uuid",
      "username": "admin",
      "first_name": "John",
      "last_name": "Doe"
    },
    "expires_at": "2026-12-31T23:59:59Z"
  },
  "warning": "Save the token_secret now - it cannot be retrieved later!"
```

```
}
```

List Tokens

Endpoint: `GET /api/v1/auto-enrollment/tokens`

Response: `200 OK`

```
[
  {
    "id": "uuid",
    "token_name": "Proxmox Production",
    "token_key": "patchmon_ae_abc123...",
    "is_active": true,
    "allowed_ip_ranges": ["192.168.1.10"],
    "max_hosts_per_day": 100,
    "hosts_created_today": 15,
    "last_used_at": "2025-10-11T14:30:00Z",
    "expires_at": "2026-12-31T23:59:59Z",
    "created_at": "2025-10-01T10:00:00Z",
    "default_host_group_id": "uuid",
    "metadata": {"integration_type": "proxmox-lxc"},
    "host_groups": {
      "id": "uuid",
      "name": "Proxmox LXC",
      "color": "#3B82F6"
    },
    "users": {
      "id": "uuid",
      "username": "admin",
      "first_name": "John",
      "last_name": "Doe"
    }
  }
]
```

Get Token Details

Endpoint: `GET /api/v1/auto-enrollment/tokens/:tokenId`

Response: `200 OK` (same structure as single token in list)

Update Token

Endpoint: `PATCH /api/v1/auto-enrollment/tokens/:tokenId`

Request:

```
{
  "is_active": false,
  "max_hosts_per_day": 200,
  "allowed_ip_ranges": ["192.168.1.0/24"],
  "expires_at": "2027-01-01T00:00:00Z"
}
```

Response: `200 OK`

```
{
  "message": "Token updated successfully",
  "token": { /* updated token object */ }
}
```

Delete Token

Endpoint: `DELETE /api/v1/auto-enrollment/tokens/:tokenId`

Response: `200 OK`

```
{
  "message": "Auto-enrollment token deleted successfully",
  "deleted_token": {
    "id": "uuid",
    "token_name": "Proxmox Production"
  }
}
```

Enrollment Endpoints (Token Authentication)

Authentication via headers:

```
X-Auto-Enrollment-Key: patchmon_ae_abc123...
X-Auto-Enrollment-Secret: def456...
```

Download Enrollment Script

Endpoint: `GET /api/v1/auto-enrollment/script`

Query Parameters:

- `type` (required): Script type (`proxmox-lxc` or `direct-host`)
- `token_key` (required): Auto-enrollment token key
- `token_secret` (required): Auto-enrollment token secret
- `force` (optional): `true` to enable force install mode

Example:

```
curl "https://patchmon.example.com/api/v1/auto-enrollment/script?type=proxmox-lxc&token_key=KEY&token_secret=SECRET&force=true"
```

Response: `200 OK` (bash script with credentials injected)

Enroll Single Host

Endpoint: `POST /api/v1/auto-enrollment/enroll`

Request:

```
{
  "friendly_name": "webserver",
  "machine_id": "proxmox-lxc-100-abc123",
  "metadata": {
    "vmid": "100",
    "proxmox_node": "proxmox01",
    "ip_address": "10.0.0.10",
    "os_info": "Ubuntu 22.04 LTS"
  }
}
```

Response: `201 Created`

```
{
  "message": "Host enrolled successfully",
  "host": {
    "id": "uuid",
    "friendly_name": "webserver",
    "api_id": "patchmon_abc123",
    "api_key": "def456ghi789",
    "host_group": {
```

```
      "id": "uuid",
      "name": "Proxmox LXC",
      "color": "#3B82F6"
    },
    "status": "pending"
  }
}
```

Error Responses:

> **Note:** The API does not perform duplicate host checks. Duplicate prevention is handled client-side by the enrollment script, which checks for an existing agent configuration inside each container before calling the API.

`429 Too Many Requests` - Rate limit exceeded:

```
{
  "error": "Rate limit exceeded",
  "message": "Maximum 100 hosts per day allowed for this token"
}
```

Bulk Enroll Hosts

Endpoint: `POST /api/v1/auto-enrollment/enroll/bulk`

Request:

```
{
  "hosts": [
    {
      "friendly_name": "webserver",
      "machine_id": "proxmox-lxc-100-abc123"
    },
    {
      "friendly_name": "database",
      "machine_id": "proxmox-lxc-101-def456"
    }
  ]
}
```

Limits:

- Minimum: 1 host
- Maximum: 50 hosts per request

Response: `201 Created`


```
{
  "message": "Bulk enrollment completed: 2 succeeded, 0 failed, 0 skipped",
  "results": {
    "success": [
      {
        "id": "uuid",
        "friendly_name": "webserver",
        "api_id": "patchmon_abc123",
        "api_key": "def456"
      },
      {
        "id": "uuid",
        "friendly_name": "database",
        "api_id": "patchmon_ghi789",
        "api_key": "jkl012"
      }
    ],
    "failed": [],
    "skipped": []
  }
}
```


FAQ

General Questions

Q: Can I use the same token for multiple Proxmox hosts?
A: Yes, as long as the combined enrollment count stays within `max_hosts_per_day` limit. Rate limits are per-token, not per-host.

Q: What happens if I run the script multiple times?
A: Already-enrolled containers are automatically skipped. The script checks for existing agent configuration inside each container and skips those where the agent is already installed and

responsive. Safe to rerun!

Q: Can I enroll stopped LXC containers?
A: No, containers must be running. The script needs to execute commands inside the container to install the agent. Start containers before enrolling.

Q: Does this work with Proxmox VMs (QEMU)?
A: No, this script is LXC-specific and uses `pct exec` to enter containers. VMs require manual enrollment or a different automation approach (SSH-based).

Q: How do I unenroll a host?
A: Go to PatchMon UI → Hosts → Select host → Delete. The agent will stop reporting and the host record is removed from the database.

Q: Can I change the host group after enrollment?
A: Yes! In PatchMon UI → Hosts → Select host → Edit → Change host group.

Q: Can I see which hosts were enrolled by which token?
A: Yes, check the host "Notes" field in PatchMon. It includes the token name and enrollment timestamp.

Q: What if my Proxmox host IP address changes?
A: Update the token's `allowed_ip_ranges` in PatchMon UI (Settings → Integrations → Edit Token).

Q: Can I have multiple tokens with different host groups?
A: Yes! Create separate tokens for prod/dev/staging with different default host groups. Great for environment segregation.

Q: Is there a way to trigger enrollment from PatchMon GUI?
A: Not currently (would require inbound network access). The script must run on the Proxmox host. Future versions may support webhooks or agent-initiated enrollment.

Security Questions

Q: Are token secrets stored securely?
A: Yes, token secrets are hashed using bcrypt before storage. Only the hash is stored in the database, never the plain text.

Q: What happens if someone steals my auto-enrollment token?
A: They can create new hosts up to the rate limit, but cannot control existing hosts or access host data. Immediately disable the token in PatchMon UI if compromised.

Q: Can I audit who created which tokens?
A: Yes, each token stores the `created_by_user_id`. View in PatchMon UI or query the database.

Q: How does IP whitelisting work?
A: PatchMon checks the client IP from the HTTP request. If `allowed_ip_ranges` is configured, the IP must match one of the allowed ranges using CIDR notation (e.g., `192.168.1.0/24`). Single IP addresses are also supported (e.g., `192.168.1.10`).

Q: Can I use the same credentials for enrollment and agent communication?
A: No, they're separate. Auto-enrollment credentials create hosts. Each host gets unique API credentials for agent communication. This separation limits the blast radius of credential compromise.

Technical Questions

Q: Why does the agent require curl inside the container?
A: The agent script uses curl to communicate with PatchMon. The enrollment script automatically installs curl if missing.

Q: What Linux distributions are supported in containers?
A: Ubuntu, Debian, CentOS, RHEL, Rocky Linux, AlmaLinux, Alpine Linux. Any distribution with apt/yum/dnf/apk package managers.

Q: How much bandwidth does enrollment use?
A: Minimal. The script download is ~15KB, agent installation is ~50-100KB per container. Total: ~1-2MB for 10 containers.

Q: Can I run enrollment in parallel for faster processing?
A: Not recommended. The script processes containers sequentially to avoid overwhelming the PatchMon server. For 100+ containers, consider the bulk API endpoint.

Q: Does enrollment restart containers?
A: No, containers remain running. The agent is installed without reboots or service disruptions.

Q: What if the container doesn't have a hostname?
A: The script uses the container name from Proxmox as a fallback.

Q: Can I customize the agent installation?
A: Yes, modify the `install_url` in the enrollment script or use the PatchMon agent installation API parameters.

Troubleshooting Questions

Q: Why does enrollment fail with "dpkg was interrupted"?
A: Your container has broken packages. Use `FORCE_INSTALL=true` to bypass, or manually fix dpkg:

```
pct enter 100
dpkg --configure -a
apt-get install -f
```

Q: Why does the agent show "pending" status forever?
A: Agent likely can't reach PatchMon server. Check:

1. Container network connectivity: `pct exec 100 -- ping patchmon.example.com`
2. Agent service running: `pct exec 100 -- systemctl status patchmon-agent.service`
3. Agent logs: `pct exec 100 -- journalctl -u patchmon-agent.service`

Q: Can I test enrollment without actually creating hosts?
A: Yes, use dry run mode: `DRY_RUN=true ./proxmox_auto_enroll.sh`

Q: How do I get more verbose output?
A: Use debug mode: `DEBUG=true ./proxmox_auto_enroll.sh`

Support and Resources

Documentation

- **PatchMon Documentation**: https://docs.patchmon.net
- **API Reference**: https://docs.patchmon.net/api
- **Agent Documentation**: https://docs.patchmon.net/agent

Community

- **Discord**: https://patchmon.net/discord
- **GitHub Issues**: https://github.com/PatchMon/PatchMon/issues
- **GitHub Discussions**: https://github.com/PatchMon/PatchMon/discussions

Professional Support

For enterprise support, training, or custom integrations:

- **Email**: support@patchmon.net
- **Website**: https://patchmon.net/support

PatchMon Team

Revision #4
Created 2025-10-11 17:35:56 UTC by Iby

Updated 2026-02-12 00:26:29 UTC by Iby