Image not found or type unknown

# Addon Module

The puq_proxmox_kvm addon module is a required companion to the server module. It provides a central dashboard, IP address pool management, DNS zone management (Cloudflare, HestiaCP, PowerDNS), a VM management view with deploy logs, and the cron task orchestration. The addon must be installed and activated for the server module to work.

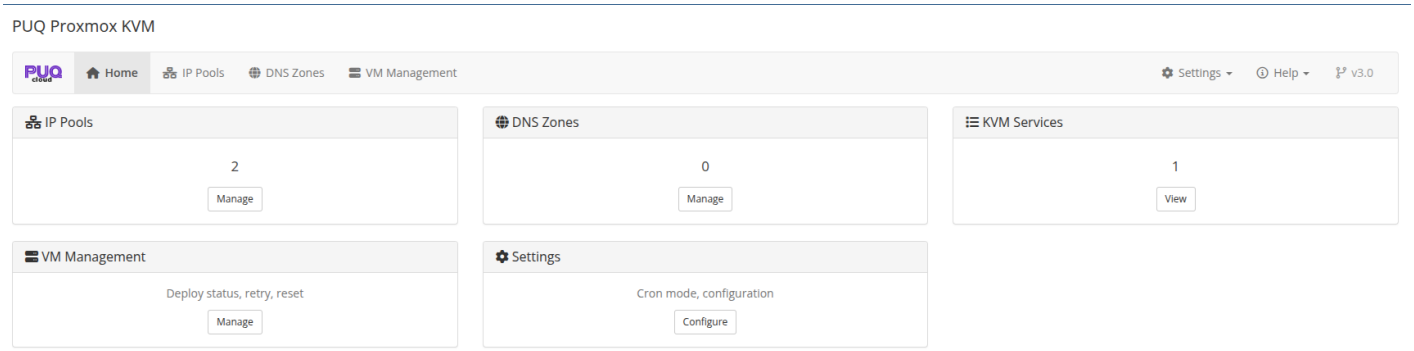

- Dashboard
- IP Pools
- DNS Zones
- VM Management
- Settings

# Dashboard

## Proxmox KVM module **WHMCS**

Order now | Download | FAQ

The addon module dashboard provides a quick overview of all managed resources.




# Dashboard Cards

| Card | Description |
|---|---|
| **IP Pools** | Total number of configured IP address pools |
| **DNS Zones** | Total number of configured DNS zones |
| **KVM Services** | Total number of active KVM services |
| **VM Management** | Link to the centralized VM management page |
| **Settings** | Link to the module settings |

# Navigation

The top navigation bar provides access to all addon sections:

- **Home** — Dashboard (this page)
- **IP Pools** — IPv4/IPv6 address pool management
- **DNS Zones** — DNS zone configuration
- **VM Management** — Centralized VM monitoring and management
- **Settings** — General settings, cron configuration
- **Help** — Links to documentation and support

# IP Pools

## Proxmox KVM module **WHMCS**

Order now | Download | FAQ

IP Pools allow you to manage blocks of IPv4 and IPv6 addresses that are automatically assigned to virtual machines during provisioning.

> **Changed in v3.0.** IP Pools are now a first-class feature of the dedicated `puq_proxmox_kvm` addon. In v1.3–v2.x the same pool management lived in the separate **PUQ Customization** addon, which is no longer required — on first activation the new addon **automatically imports** all pools from the legacy `puq_customization_ip_pools` tables, including their allocations and per-server assignments. If you are migrating from an older version you do not need to recreate the pools by hand.

> **Legacy alternative (still supported).** If you prefer not to use pools at all, you can still define the IP addresses directly on the WHMCS server entry using the pipe-delimited **Assigned IP Addresses** format — see Create new server for Proxmox in WHMCS. The server module will pick an IP from whichever source has free entries (pools first, then the legacy list).

# IP Pools List

Navigate to **Addons > PUQ Proxmox KVM > IP Pools** to view all configured pools.

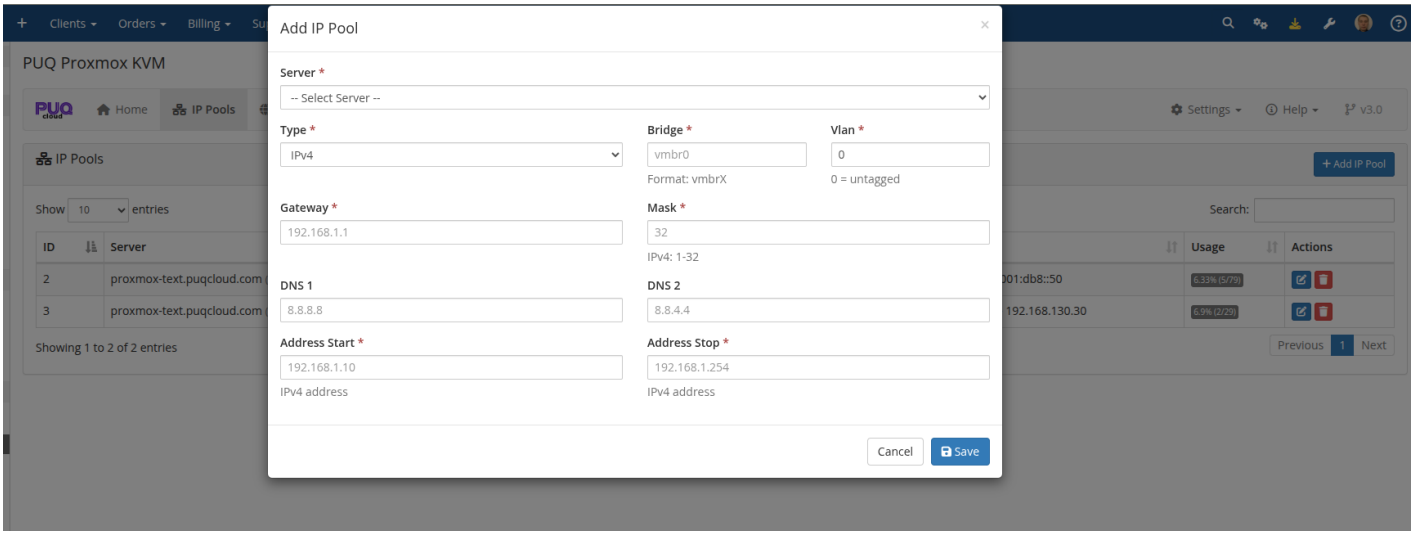

> **New in v3.2.** The Addresses column now shows the ready-made **rDNS zone name** that corresponds to each pool's prefix — copy it directly into the DNS Zones form when you want reverse DNS for that pool's IPs. No need to compute nibble reversals by hand. Both IPv4 (`/8`, `/16`, `/24`) and IPv6 (any nibble-aligned prefix) are supported.

The table displays:

| Column | Description |
|---|---|
| **ID** | Pool identifier |
| **Server** | Associated Proxmox server |
| **Type** | IPv4 or IPv6 |
| **Bridge** | Network bridge (e.g., `vmbr0`) |
| **Vlan** | VLAN tag (0 = no VLAN) |
| **Gateway** | Default gateway address |
| **Mask** | Subnet mask |
| **Addresses** | Total IPs in pool |
| **Usage** | Visual bar showing allocated vs available |
| **Actions** | Edit / Delete buttons |

# Adding an IP Pool

Click **+ Add IP Pool** to open the creation dialog.

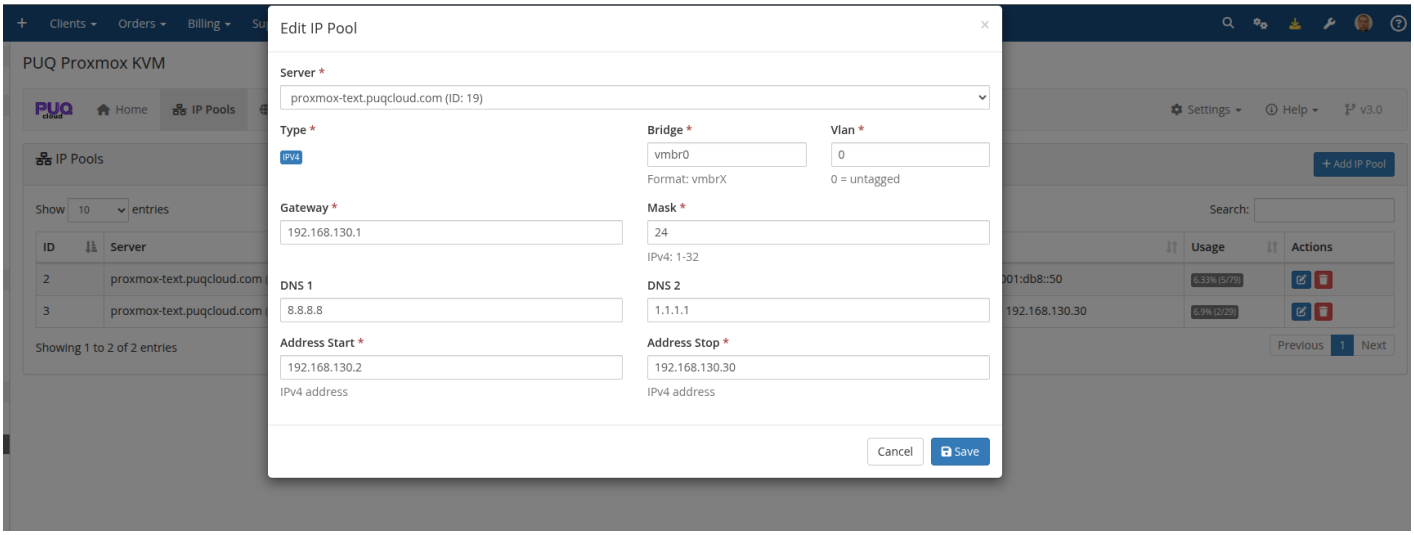



Fill in the following fields:

| Field | Description | Example |
|---|---|---|
| **Server** | Select the Proxmox server | `pve-waw1` |
| **Type** | IPv4 or IPv6 | `IPv4` |
| **Bridge** | Network bridge name | `vmbr0` |
| **Vlan** | VLAN tag (0 for untagged) | `0` |
| **Gateway** | Default gateway address | `192.168.130.1` |
| **Mask** | Subnet mask (1-32 for IPv4, 1-128 for IPv6) | `24` |
| **DNS 1** | Primary DNS server | `8.8.8.8` |
| **DNS 2** | Secondary DNS server | `1.1.1.1` |
| **Address Start** | First IP in the range | `192.168.130.2` |
| **Address Stop** | Last IP in the range | `192.168.130.254` |

# Editing an IP Pool

Click the **Edit** button next to any pool to modify its settings.




> **Note:** Modifying a pool does not affect already-assigned IP addresses. Changes only apply to new allocations.

# IP Allocation Process

IPs are automatically allocated from pools during VM provisioning when:

1. The server has no assigned IPs configured in WHMCS server settings
2. The addon module is installed and activated
3. The product's network configuration has **Auto bridge/VLAN** enabled

The system selects IPs from pools matching the server associated with the product. IPv4 and IPv6 addresses are allocated from separate pools.

# Validation Rules

- Bridge must be a valid Proxmox bridge name
- Gateway must match the pool type (IPv4 for IPv4 pools, IPv6 for IPv6 pools)
- Server must be a valid Proxmox server configured in WHMCS
- Address range must be valid for the given type

# DNS Zones

## Proxmox KVM module **WHMCS**

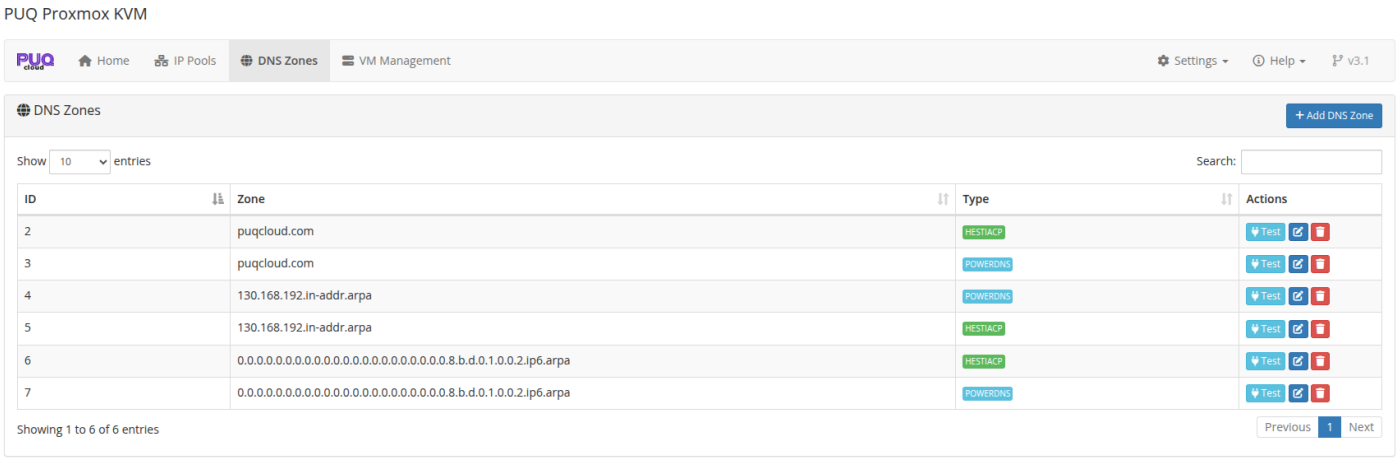
Order now | Download | FAQ

DNS Zones enable automatic management of forward (A/AAAA) and reverse (PTR) DNS records for every virtual machine the module provisions. Configure your zones once and the module takes care of creation on deploy, refresh on package change, and cleanup on termination — across all three supported providers.

> **Changed in v3.2.** Added native PowerDNS support, asynchronous DNS record creation, live cron output, and automatic reverse-zone hints on the IP Pools page. DNS errors are fully non-blocking — a misconfigured or unreachable provider never stops deployment, package change, or termination. Credentials are no longer echoed back to the browser.

# DNS Zones list

Navigate to **Addons → PUQ Proxmox KVM → DNS Zones**.




Each zone line shows: internal ID, the zone name, the provider type badge, and per-row actions ( **Test** connection, **Edit**, **Delete**).

You can add any number of zones. When a VM is deployed, the module matches the VM's FQDN and every assigned IP against all configured zones and writes to **every** matching zone.

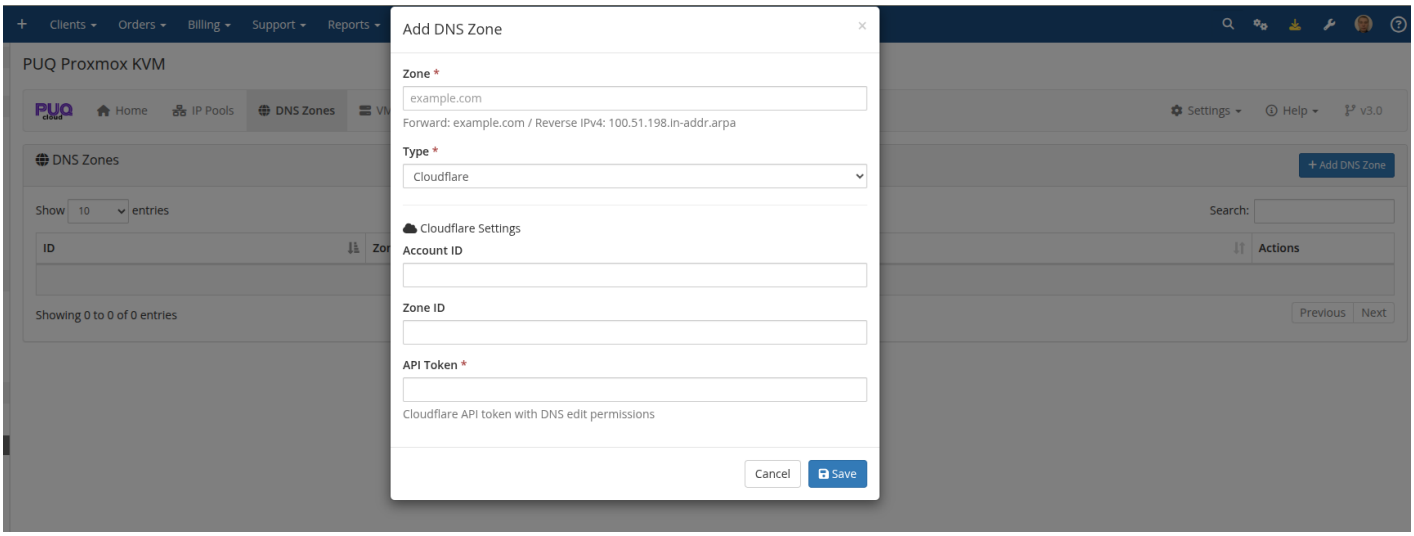
# Supported providers

The module supports three DNS providers. You can mix them freely — forward zones on Cloudflare, reverse zones on PowerDNS, legacy zones on HestiaCP, all at the same time.

| Provider | Forward (A/AAAA) | Reverse (PTR) | API style |
|---|---|---|---|
| **Cloudflare** | yes | yes | REST v4, bearer token |
| **HestiaCP** | yes | yes | custom CLI-over-HTTP, admin user + password |
| **PowerDNS** | yes | yes | Authoritative Server REST API, `X-API-Key` |

# Adding a DNS zone

Click **+ Add DNS Zone**. Choose the provider in the Type dropdown; the form fields change to match the provider.




# Cloudflare

| Field | Description |
|---|---|

| | |
|---|---|
| **Zone** | The zone name as it appears in Cloudflare (e.g., `example.com` for forward, `130.168.192.in-addr.arpa` for reverse). |
| **Type** | `Cloudflare` |
| **Account ID** | Cloudflare Account ID from the Cloudflare dashboard. |
| **Zone ID** | Cloudflare Zone ID from the zone's Overview page. |
| **API Token** | Cloudflare API token scoped to DNS edit on this zone. |

# HestiaCP

| Field | Description |
|---|---|
| **Zone** | Domain name as configured on the HestiaCP server. |
| **Type** | `HestiaCP` |
| **Server URL** | HestiaCP URL, e.g., `https://hestia.example.com:8083/`. The trailing slash is added automatically if omitted. |
| **Admin User** | HestiaCP admin username. |
| **Admin Password** | HestiaCP admin password. |
| **User** | HestiaCP user that owns the DNS zone. |

# PowerDNS

| Field | Description |
|---|---|
| **Zone** | Zone name as configured in PowerDNS (`example.com.` for forward, `8.b.d.0.1.0.0.2.ip6.arpa.` for IPv6 reverse). Trailing dots are normalized automatically. |
| **Type** | `PowerDNS` |
| **Server URL** | PowerDNS REST API base URL, e.g., `https://pdns.example.com:8081`. |
| **API Key** | Value of the `X-API-Key` header configured in `pdns.conf`. |

Make sure the PowerDNS API is enabled in your `pdns.conf`:

```
api=yes
api-key=<your-key-here>
webserver=yes
webserver-address=0.0.0.0
webserver-port=8081
```

```
webserver-allow-from=127.0.0.1,<whmcs-ip>
```

PowerDNS uses RRset-based updates: records are added or replaced atomically per name+type. PTR/CNAME/NS content is automatically wrapped with a trailing dot to satisfy PowerDNS strict validation.

# Forward vs reverse zones

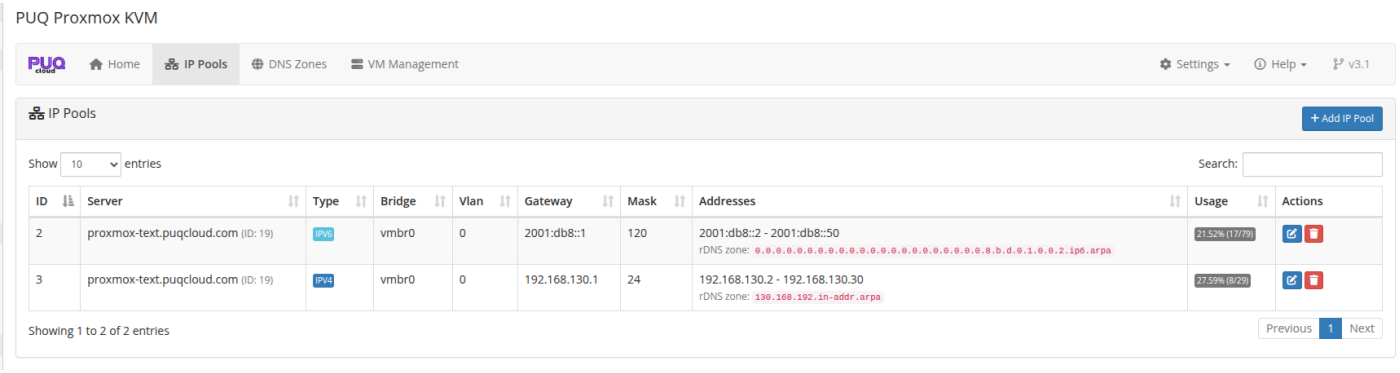
The module does not distinguish between "forward" and "reverse" zones in the UI — there is just one **Zone** field. What makes a zone forward or reverse is simply its **name**:

- **Forward zone**: a regular domain name. Example: `puqcloud.com`.
  - Stores A records (IPv4 → hostname) and AAAA records (IPv6 → hostname).
  - Needed for clients to reach their VM by DNS name.
- **Reverse IPv4 zone**: ends in `.in-addr.arpa`. Example: `130.168.192.in-addr.arpa` (covers the `192.168.130.0/24` network).
  - Stores PTR records (IP → hostname).
  - Needed for outbound mail, rDNS verification, PTR lookups.
- **Reverse IPv6 zone**: ends in `.ip6.arpa`. Example: `0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.8.b.d.0.1.0.0.2.ip6.arpa` (covers `2001:db8::/120`).
  - Stores PTR records for IPv6.

You can add **multiple zones of different providers with the same name**. For example, if you run a primary PowerDNS and a secondary HestiaCP, add two zones: `puqcloud.com / PowerDNS` and `puqcloud.com / HestiaCP`. The module will push every forward record to both on deploy and remove from both on termination.

# Finding the reverse-zone name for a pool

You don't have to compute the reverse zone name for an IP pool by hand. The addon does it for you on the IP Pools page: the required **rDNS zone** for each pool is shown on the second line in the Addresses column, and live in the add/edit modal as you type the prefix:




Example from the screenshot:

- Pool `192.168.130.2 - 192.168.130.30` (gateway `192.168.130.1`, mask `/24`) → **rDNS zone `130.168.192.in-addr.arpa`**
- Pool `2001:db8::2 - 2001:db8::50` (gateway `2001:db8::1`, mask `/120`) → **rDNS zone `0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.8.b.d.0.1.0.0.2.ip6.arpa`**

Copy that value straight into the **Zone** field when adding a DNS zone for reverse records. The pool itself doesn't need any DNS configuration — the module only uses this name to know where PTR records should go, and only if such a zone actually exists in DNS Zones.

# Prefix alignment

- **IPv4** reverse zones work naturally for `/8`, `/16`, `/24` prefixes. Non-octet-aligned prefixes (e.g., `/22`, `/28`) require "classless delegation" — a more advanced DNS setup described in RFC 2317 — which is beyond the module's scope. Pools with such prefixes show `classless delegation required` instead of a ready-made zone name.
- **IPv6** reverse zones work for any nibble-aligned prefix (multiple of 4): `/4`, `/8`, `/12`, `/16`, … `/124`, `/128`. The module supports the full range.

# How DNS automation works

When a VM transitions through state changes, the following DNS operations run automatically:

| Event | Forward records | Reverse records |
|---|---|---|
| **Deploy** (state `clone → set_dns`) | Create A and/or AAAA for `<vmname>.<domain>` | Create PTR for every assigned IP |
| **Change package** (state `change_package → cp_update_ip`) | Delete all, then recreate — reflects any IP changes | Delete all, then recreate |
| **Terminate** | Delete forward records for the VM's FQDN | Delete PTR records for everyassigned IP |
| **Set DNS records admin button** | Delete + recreate — forces a full resync | Delete + recreate |

The main domain used for forward records comes from product configuration: **Admin → Products → [your product] → Module Settings → Integrations → Main domain**. For example if the main domain is `puqcloud.com` and the VM's internal name is `5551-1776530141`, the FQDN registered in DNS is `5551-1776530141.puqcloud.com`.

# Zone matching

For a given DNS operation the module walks every configured zone and checks whether the record name would fit that zone:

- A forward `vm-123.puqcloud.com` matches any zone whose name is a suffix: `puqcloud.com`, for example. It does not match `puq.com` or `example.com`.
- A reverse PTR `10.1.168.192.in-addr.arpa` matches `1.168.192.in-addr.arpa`, `168.192.in-addr.arpa`, or `192.in-addr.arpa` — any level of reverse delegation.
- An IPv6 PTR matches any `*.ip6.arpa` zone that is a proper suffix of the full 32-nibble reverse name.

Zones that don't match a given VM's records are simply skipped — there's no error.

# Non-blocking errors

Every per-zone and per-record operation is wrapped in its own try/catch. If a zone's provider is down, credentials are wrong, or a specific record creation fails:

- The error is logged to **Utilities → Logs → Module Log** with full context.
- A live cron output shows `fwd ERR` / `rev ERR` for that specific operation.
- **Deploy / change package / terminate continues** — the next zone, the next record, and the next pipeline step all run.
- When errors occur, a summary entry is written to the WHMCS module log so admins can audit failures after the fact.

This is by design: a DNS outage must not block a client from getting their VM. You can always run **Set DNS records**

later from the admin service page once the DNS provider is back online (see below).

# Set DNS records admin button

On a service's admin page in WHMCS, the module exposes a **Set DNS records** button under Module Commands. It performs a full DNS resync for that specific VM: delete every existing forward and reverse record, then recreate them from the VM's current IPs and domain.

Starting with v3.2 this runs **asynchronously**. Clicking the button queues the job (sets `vm_status = 'set_dns_records'`) and returns immediately. The next cron tick picks up the VM and runs the full delete + create cycle with live output — useful for services with dozens of reverse records where the synchronous version used to time out.

The progress shows up in VM Management → Log modal and in the cron stdout just like during deploy.

# Credentials never leave the server

In v3.2 the DNS Zones list API masks all secret fields (API token, admin password, API key) with a `__KEEP__` sentinel before sending data to the browser. The edit form shows `(unchanged — enter new to replace)` placeholders:

- If you **don't type anything** in a secret field on save, the stored value is kept.
- If you **type a new value**, it overwrites the stored value.

This means tokens cannot be stolen by inspecting the edit form's HTML or by a compromised admin browser. The only way to read a stored credential is direct database access.

# Testing a zone

The **Test** button per row runs a live connectivity and authorization check against the provider. For Cloudflare and PowerDNS it fetches the zone metadata; for HestiaCP it issues a zone listing call.

A green toast means the provider is reachable with valid credentials and the zone name matches what's configured on the server. A red toast shows the exact error returned by the provider.

---

# Legacy DNS endpoint (`dns.php`)

> **Still supported.** The legacy read-only JSON endpoint introduced in v1.4 is kept for backwards compatibility with external DNS automations. It does not write to any DNS server — it just returns the current forward/reverse mapping so you can feed it into your own DNS-sync script.

Send a `GET` request to:

```
https://<WHMCS-SERVER>/modules/servers/puqProxmoxKVM/lib/dns/dns.php
```

Example response:

```
[
   {
      "forward": "vlan-1-4779.vps.uuq.pl",
      "ip": "192.168.0.2",
      "reverse": "mail.uuq.pl"
   },
   {
      "forward": "vps-1-4780.vps.uuq.pl",
      "ip": "192.168.0.3",
      "reverse": "test.vps.uuq.pl"
   }
]
```

# Access control

Restrict access with `.htaccess` next to the file:

```
order deny,allow
deny from all
allow from <allowed_IP_address>
```

> For new integrations, use the native DNS providers (Cloudflare / HestiaCP / PowerDNS) instead of scraping this endpoint. The native integration handles forward + reverse, deletion on terminate, retry on transient errors, credential masking, and produces a live audit trail in the cron log and module log.

# Related reading

- IP Pools — where the rDNS zone name is computed for you.
- Deploy Process — when forward and reverse records are created.
- Change Package — DNS refresh on package changes.
- Terminate Process — DNS cleanup on service termination.

# VM Management

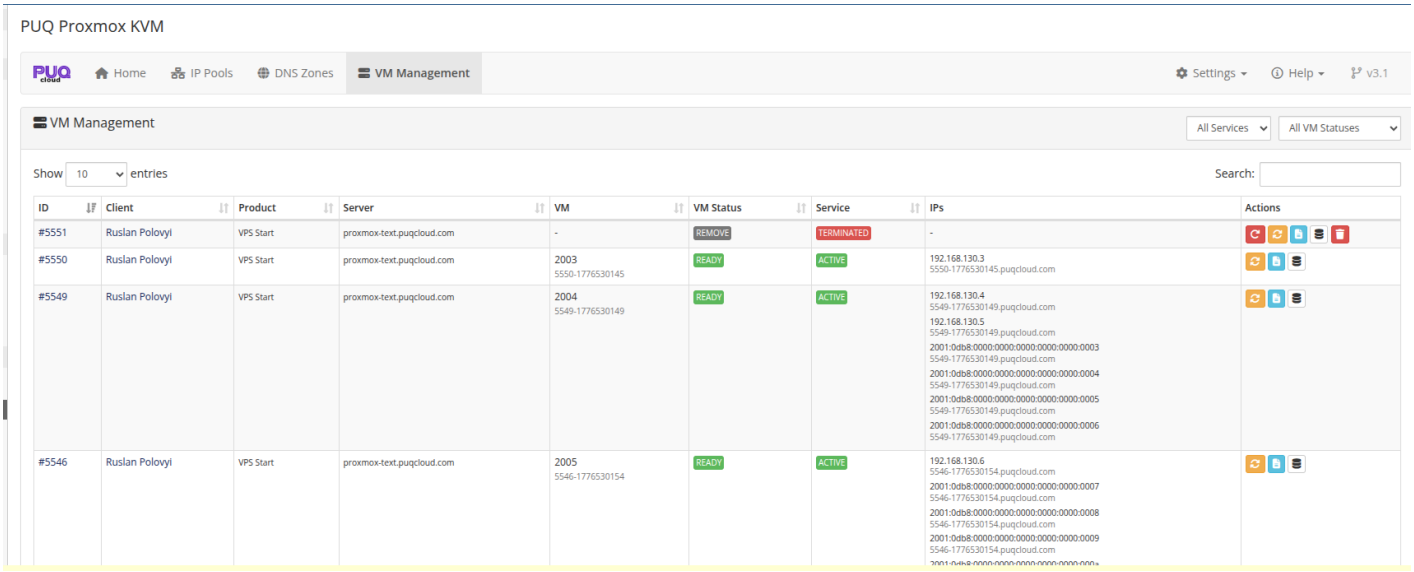
## Proxmox KVM module **WHMCS**

Order now | Download | FAQ

The VM Management page provides a centralized view of every KVM virtual machine across every Proxmox server — their current status, assigned IPs with reverse DNS, deployment history, and per-VM admin actions.

# VM list

Navigate to **Addons → PUQ Proxmox KVM → VM Management**.




The list is server-side-paginated with search and sorting. Two dropdown filters above the table narrow the view by WHMCS service status and by VM state; both remember your last choice in the browser, so the list opens the way you left it next time.

# Columns

| Column | Description |
| --- | --- |

| | |
|---|---|
| **ID** | WHMCS service ID (click for client services page). |
| **Client** | Client name with a link to the client profile. |
| **Product** | Product / plan name. |
| **Server** | Proxmox server name. |
| **VM** | Proxmox VM ID and internal VM name. |
| **VM Status** | Module lifecycle status — see status reference below. |
| **Service** | WHMCS service status: Active, Suspended, Terminated, Cancelled, Pending. |
| **IPs** | Every assigned IPv4 / IPv6 address paired with its current reverse DNS name on the line below. |
| **Actions** | Per-VM admin buttons: Redeploy, Reset, Log, DB Record, (Delete Record when applicable). |

# IPs column — IPs with rDNS

Starting with v3.2, each IP is shown together with its rDNS on a second, smaller line:

```
192.168.130.2
  5546-1776530141.puqcloud.com

2001:0db8:0000:0000:0000:0000:0000:0007
  5546-1776530141.puqcloud.com
```

Visual grouping makes it easy to scan. IPs without a rDNS entry simply don't have the second line.

# Administrative actions

| Button | Visible when | What it does |
|---|---|---|
| **Redeploy** (red, circular-arrow) | `vm_status != ready` | Destroys the VM on Proxmox, clears IPs, resets logs, sets state back to `creation` — the full deploy pipeline runs from scratch on the next cron tick. **Destructive.** |
| **Reset** (yellow, sync) | always | Opens the Reset VM Status modal — switch the VM to any of the re-runnable states. See below. |

| Button | Visible when | What it does |
|---|---|---|
| **Log** (blue, file) | always | Opens the VM Log modal with per-run and per-step history. |
| **DB Record** (grey, database) | always | Opens the raw `puqProxmoxKVM_vm_info` row for inspection and manual editing. Last-resort troubleshooting tool. |
| **Delete Record** (red, trash) | `vm_status in (error_terminate, remove)` | Removes the row from `puqProxmoxKVM_vm_info`. Does **not** touch Proxmox or `tblhosting`. Confirmation dialog warns explicitly. |

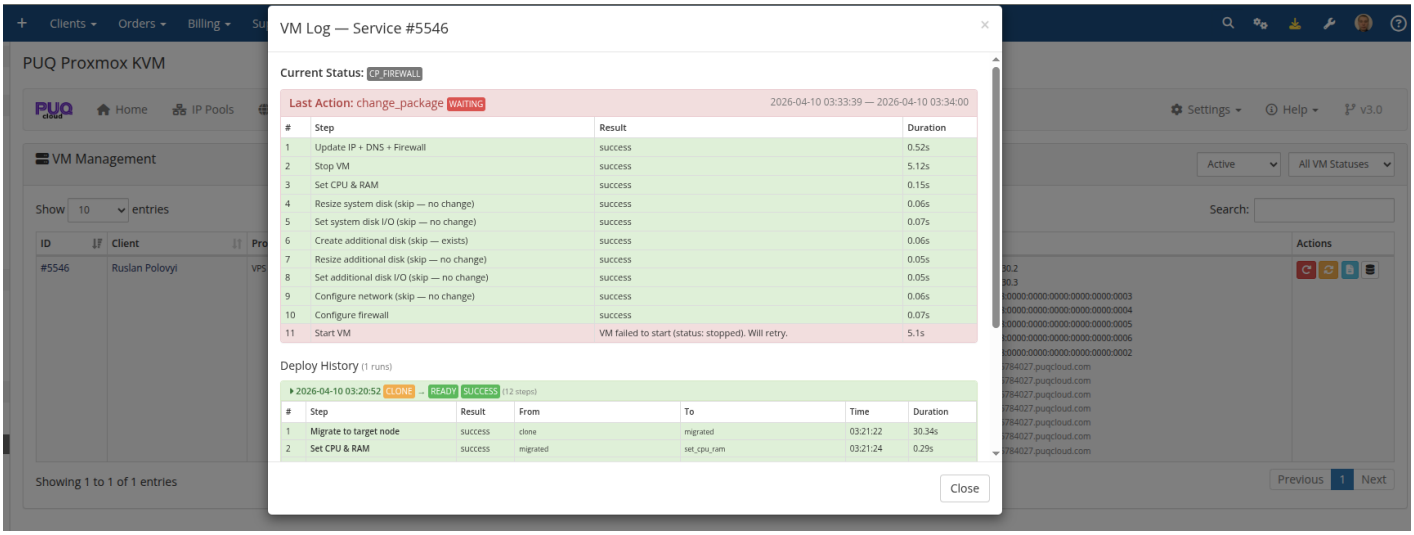
# Reset VM Status

The Reset modal lets you switch a VM to any of the re-runnable states. An embedded reference table explains when each one is appropriate:

| Status | Use case |
|---|---|
| `ready` | Return the VM to the normal "everything is fine" state. Use after you've finished a manual fix and want cron to stop touching it. |
| `creation` | Retry deploy from the beginning — typically after fixing the underlying reason an earlier deploy failed. |
| `set_ip` | Retry only the IP allocation step. |
| `change_package` | Rerun the full package-change flow. |
| `set_dns_records` | Queue a full DNS resync (delete + recreate all records). Fast and safe. |
| `terminate` | Retry termination after an `error_terminate`. |
| `remove` | Force-mark the VM as removed (no Proxmox contact). Use only when the VM is already gone from Proxmox and you just need WHMCS to stop showing it as active. |

# VM Log modal

The Log modal shows every pipeline run — deploy, change package, set DNS records, terminate — with per-step duration, result, and any errors. The most recent 50 runs are kept.




When the last run failed, a red banner at the top shows the failing action and error message. Each step row shows:

- Step label (human-readable).
- State transition (e.g., `set_ip → clone`).
- Result: `success` / `waiting` / `error: …`.
- Duration in seconds.
- Timestamp.

Skipped steps in change package (see Change Package) show `skip (no change)` and contribute zero duration.

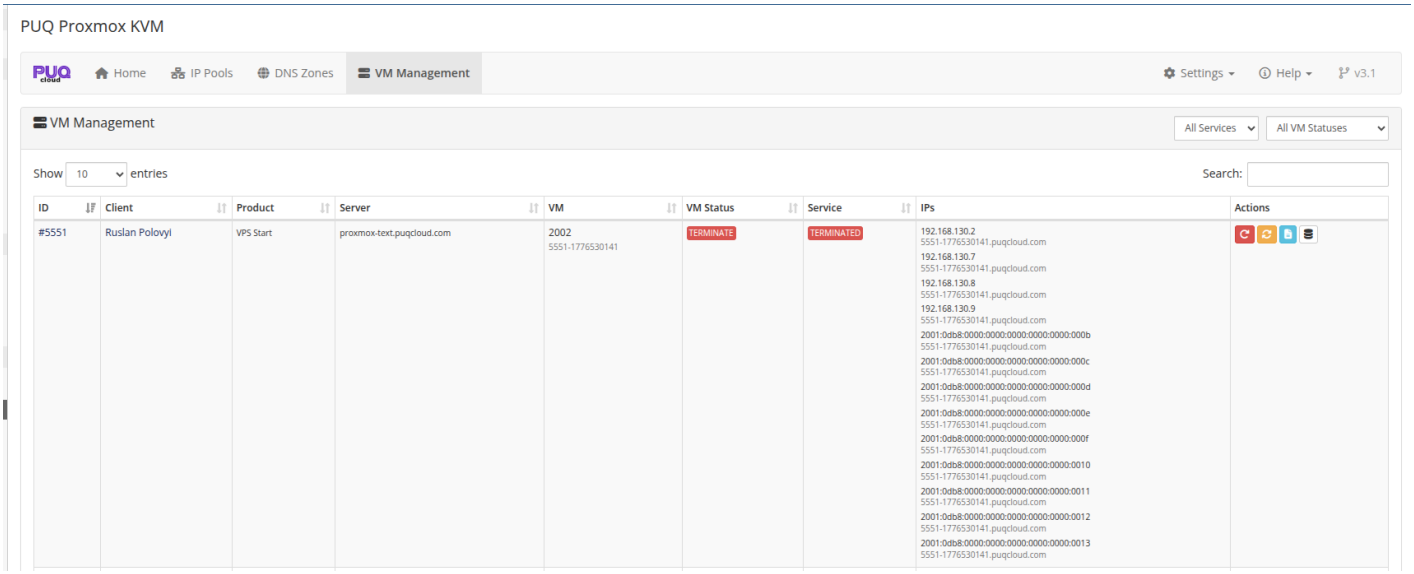
# VM status reference

| Status | Meaning | Cron behavior |
|---|---|---|
| `creation`, `set_ip`, `clone`, `set_dns`, `migrated`, `set_cpu_ram`, `set_system_disk_size`, `set_system_disk_bandwidth`, `set_created_additional_disk`, `set_additional_disk_size`, `set_additional_disk_bandwidth`, `set_network`, `set_firewall`, `set_cloudinit`, `starting` | Deploy pipeline in progress at the named step. | Cron executes the next step each tick. |
| `ready` | VM is live and the client has access. | Cron ignores. |

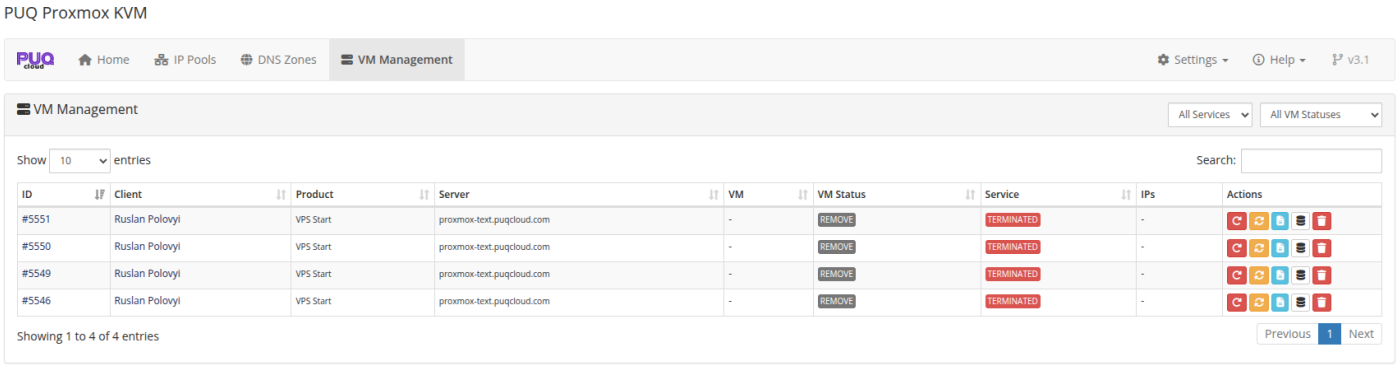
| Status | Meaning | Cron behavior |
| --- | --- | --- |
| `change_package`, `cp_update_ip`, `cp_stop`, `cp_cpu_ram`, `cp_system_disk_size`, `cp_system_disk_bandwidth`, `cp_additional_disk`, `cp_additional_disk_size`, `cp_additional_disk_bandwidth`, `cp_network`, `cp_firewall`, `cp_start` | Change-package pipeline in progress. | Cron executes the next step each tick. |
| `reinstall` | Reinstall requested; needs the existing VM removed before reverting to `set_ip`. | Cron converts to `set_ip` after removing the old VM. |
| `set_dns_records` | Queued DNS resync. | Cron does a full delete+create cycle then returns to `ready`. |
| `terminate` | Termination queued. | Cron performs stop → backups → DNS → DELETE → cleanup. |
| `error_terminate` | Terminate failed. **Admin action required.** | Cron skips. Fix the cause and reset to `terminate` or `remove`. |
| `remove` | VM has been cleaned up. | Cron skips. Optionally use **Delete Record** to remove the row. |

# Watching an async action in VM Management

After clicking a long-running action (Terminate, Set DNS records), the VM row reflects the current state badge in real time. You can watch status changes by refreshing the page or by tracking the cron standalone output:

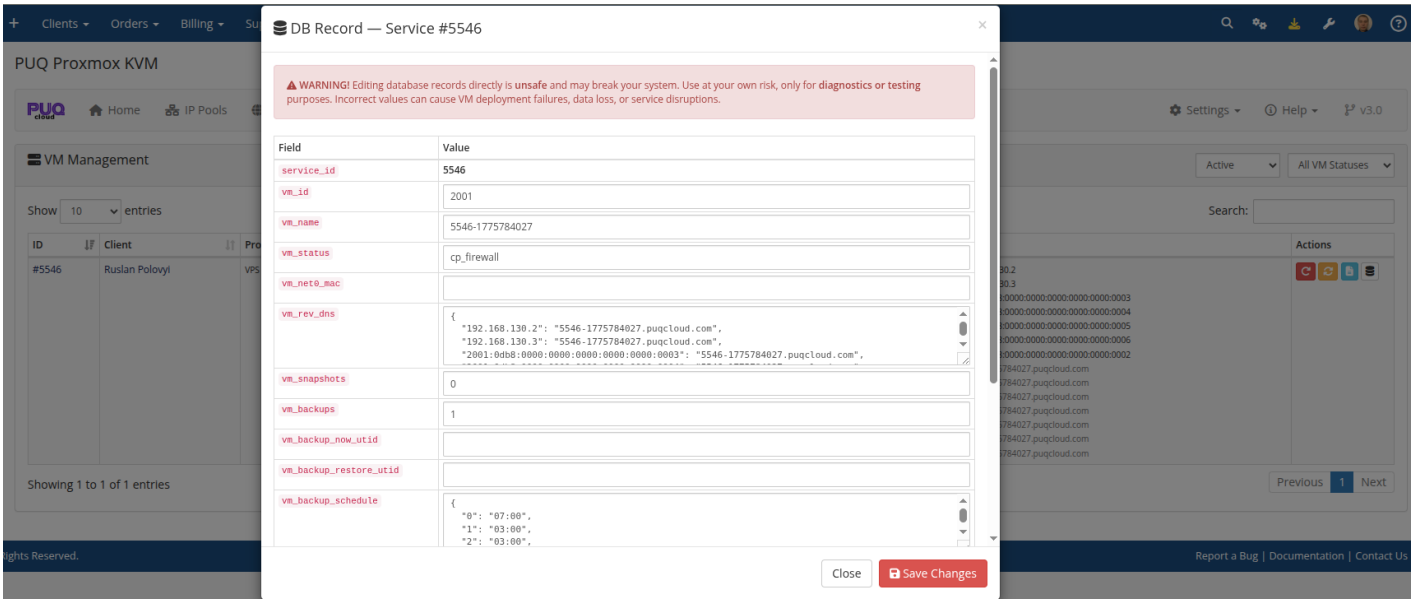



Once the cron finishes, the row appears with the final state — `remove` on success, `error_terminate` on failure:




# DB Record editor

For advanced troubleshooting, click **DB Record** to view and edit the raw `puqProxmoxKVM_vm_info` row:




> **Warning:** Direct database editing bypasses every safeguard in the state machine. Incorrect values cause deployment failures, incorrect IP accounting, or data loss. Use only when you know exactly what you're doing and the usual Reset / Redeploy actions cannot help.

# Related reading

- Deploy Process — deploy state machine driving `creation → … → ready`.
- Change Package — the `change_package → … → ready` flow.
- Terminate Process — async terminate, `error_terminate` path and recovery.
- DNS Zones & Integration — what runs when you click Set DNS records or when `set_dns_records` is queued.

# Settings

## Proxmox KVM module **WHMCS**

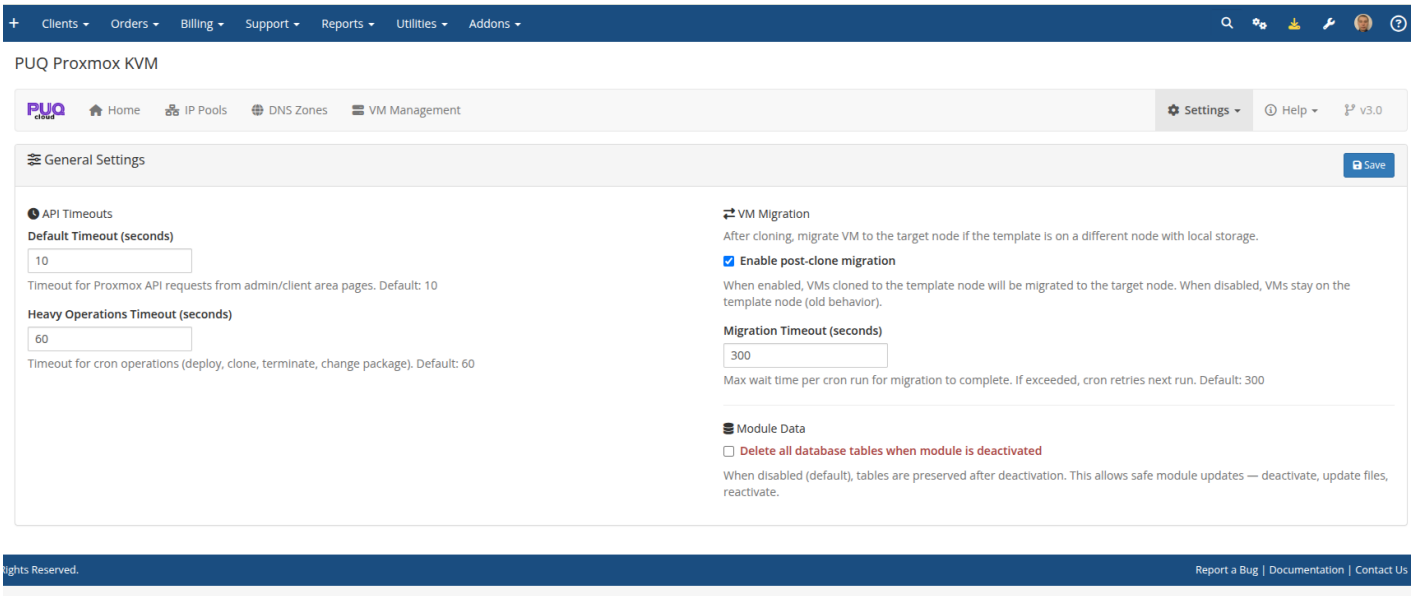
Order now | Download | FAQ

The Settings section is divided into two pages: **General** and **Cron**.

---

# General Settings

Navigate to **Addons > PUQ Proxmox KVM > Settings > General**.




# API Timeouts

| Setting | Default | Range | Description |
|---|---|---|---|
| **Default Timeout** | 10s | 1–300 | Timeout for Proxmox API requests from admin/client area pages |

| Setting | Default | Range | Description |
|---|---|---|---|
| **Heavy Operations Timeout** | 60s | 1–600 | Timeout for cronoperations (deploy, clone,terminate, changepackage) |

# VM Migration

| Setting | Default | Description |
|---|---|---|
| **Enable post-clone migration** | Yes | When enabled, VMs cloned to the template node will be automatically migrated to the target node with the correct storage |
| **Migration Timeout** | 300s | Maximum wait time per cron run for migration to complete. If exceeded, cron retries on next run |

# Module Data

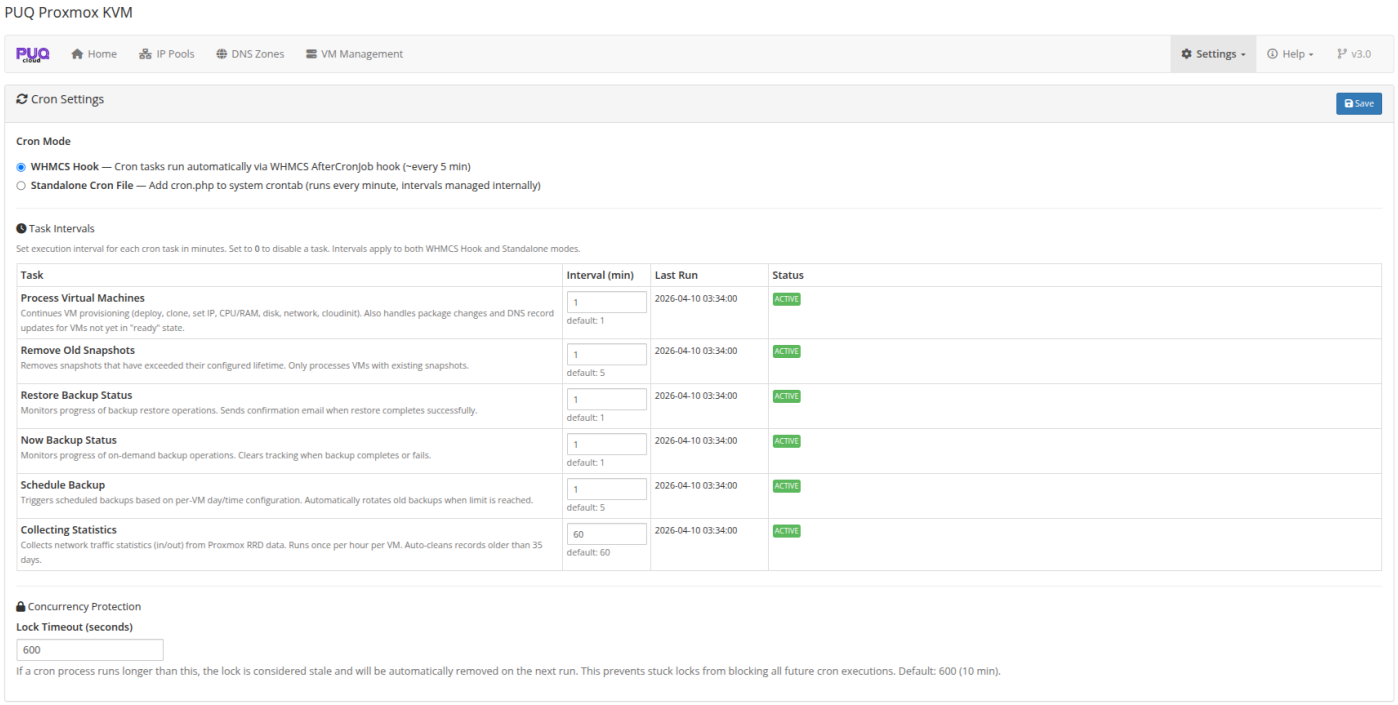
| Setting | Default | Description |
|---|---|---|
| **Delete all database tables** | No | When enabled, all module database tables are dropped on addon deactivation. When disabled (default), tables are preserved for safe updates |

---

# Cron Settings

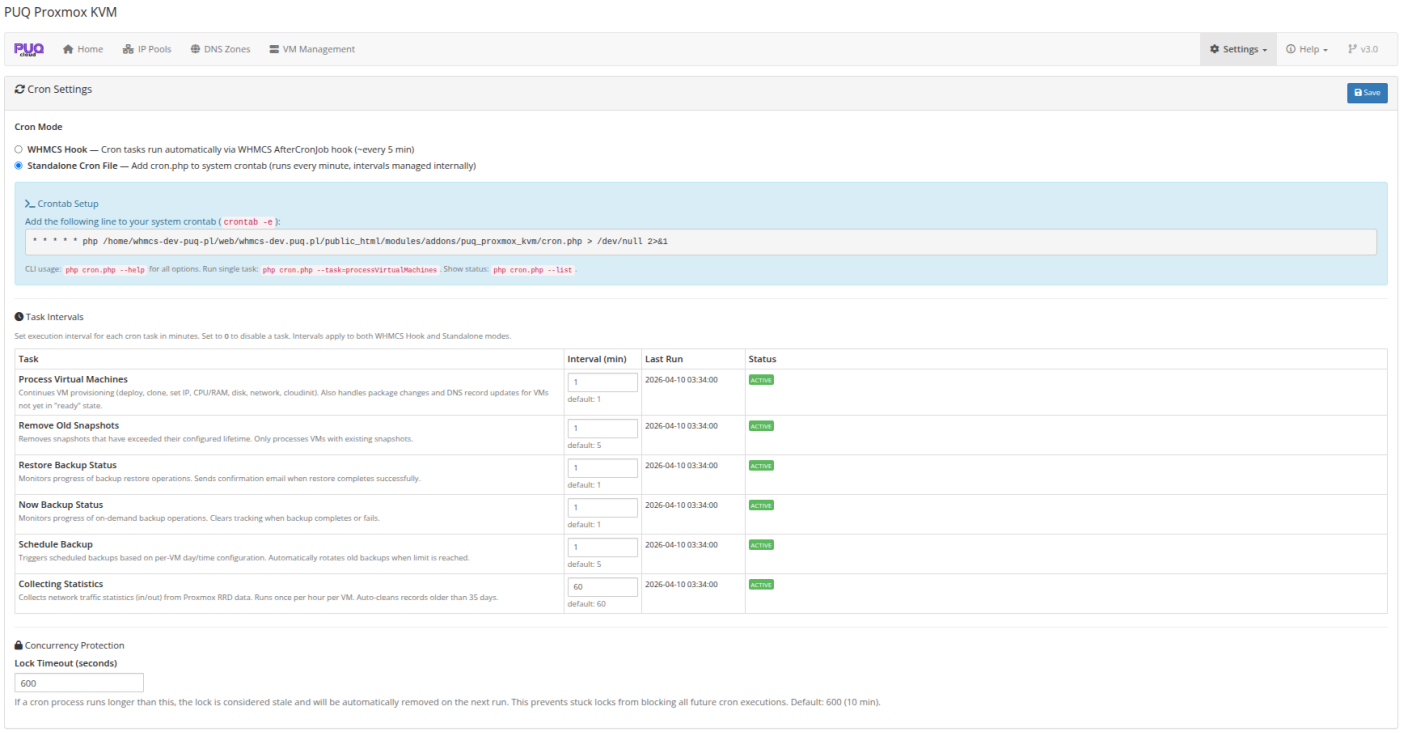
Navigate to **Addons > PUQ Proxmox KVM > Settings > Cron**.

# WHMCS Hook Mode




In this mode, cron tasks run automatically as part of the WHMCS system cron. No additional configuration is needed.

# Standalone Mode




In standalone mode, you run the cron file directly via system crontab:

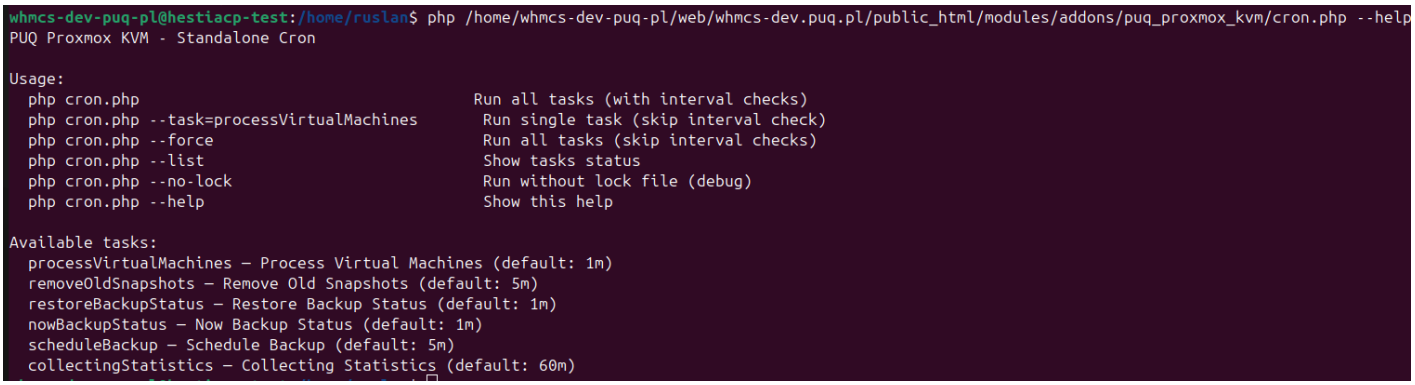
```
* * * * * php /path/to/whmcs/modules/addons/puq_proxmox_kvm/cron.php
```

# Cron Tasks

| Task | Default Interval | Description |
|---|---|---|
| **Process Virtual Machines** | 1 min | Deploys new VMs, processes change_package, refreshes DNS |
| **Remove Old Snapshots** | 5 min | Deletes snapshots past their configured lifetime |
| **Restore Backup Status** | 1 min | Checks completion of backup restore operations |
| **Now Backup Status** | 1 min | Checks completion of manual backup operations |
| **Schedule Backup** | 5 min | Runs scheduled automatic backups |
| **Collecting Statistics** | 60 min | Collects network usage metrics for billing |

- Set interval to **0** to disable a task
- **Lock Timeout** — maximum time a cron lock is held before considered stale (default: 600s)

# CLI Tools




The standalone cron file supports command-line arguments:

```
# Run all tasks
php cron.php

# Run specific task
php cron.php --task=processVirtualMachines
```

```
# Force run (ignore intervals)
php cron.php --force

# List tasks and their status
php cron.php --list

# Show help
php cron.php --help
```