

# Deploy Process

## Proxmox KVM module **WHMCS**

[Order now](#) | [Download](#) | [FAQ](#)

## Overview

When a new virtual machine is provisioned (from a WHMCS order, an admin **Create** action, or the WHMCS API), the module does **not** try to do everything inside the HTTP request. Instead it just writes the VM record with status `creation` and returns immediately — WHMCS sees the order as "accepted" within milliseconds. The actual work is done by the cron task **Process VMs** as a 15-step state machine. Each step is idempotent and resumable: if anything fails, the VM stays at its current step and the next cron tick retries it.

This design makes the module resilient to:

- Proxmox API timeouts and slow storage operations
- Cluster-wide task queue back-pressure
- Transient network failures between the WHMCS host and Proxmox
- Long-running operations (disk resize, full clones, cross-node migration)

## Deploy Pipeline

The pipeline progresses through the following states:

```
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 → ready
```

A state name represents the **last completed step** — not the current action. When a VM is in state `|set_ip|`, the IP has been allocated and the next action to run is "Clone VM".

## Step descriptions

| State (done)                                 | Next action             | What happens   |
|--|-------------------------|--|
| <code> creation </code>                      | Allocate IP             | Choose an IP pool matching the server/bridge/VLAN, reserve IPv4 and/or IPv6 addresses, write them into <code> tblhosting </code> and the VM record.  |
| <code> set_ip </code>                        | Clone VM                | Clone the template to the target storage. Supports both linked and full clones depending on product config.  |
| <code> clone </code>                         | Configure DNS records   | Create forward A/AAAA for the VM's FQDN and PTR records for all assigned IPs. Runs against every matching DNS zone (see <a href="#">DNS Zones &amp; Integration</a> ). <b>DNS errors never block deployment</b> — they are logged and the pipeline moves on. |
| <code> set_dns </code>                       | Migrate to target node  | If the template lives on a different Proxmox node than the target, do an offline migration with <code> targetstorage </code> mapping. Skipped when source and target nodes are the same.   |
| <code> migrated </code>                      | Set CPU & RAM           | Apply the cores / sockets / memory from the product configuration.   |
| <code> set_cpu_ram </code>                   | Resize system disk      | Expand the system disk to the configured size.   |
| <code> set_system_disk_size </code>          | Set system disk I/O     | Apply <code> iops_rd </code> / <code> iops_wr </code> / <code> mbps_rd </code> / <code> mbps_wr </code> bandwidth limits.  |
| <code> set_system_disk_bandwidth </code>     | Create additional disk  | If the product has an additional disk, create it on the configured storage.  |
| <code> set_created_additional_disk </code>   | Resize additional disk  | Expand it to the configured size.  |
| <code> set_additional_disk_size </code>      | Set additional disk I/O | Apply bandwidth limits to the additional disk.   |
| <code> set_additional_disk_bandwidth </code> | Configure network       | Set bridge, VLAN, rate limit, and enable the firewall flag on the NIC.   |
| <code> set_network </code>                   | Configure firewall      | Apply per-product firewall options (enable, DHCP/NDP, MAC filter, IP filter, log levels), policies, and anti-spoofing IPSet with the allocated IPs.  |

| State (done)  | Next action            | What happens  |
|---------------|------------------------|---|
| set_firewall  | Configure cloud-init   | Push hostname, IP addresses, gateway, DNS servers, username, password and SSH keys into the cloud-init drive. |
| set_cloudinit | Start VM               | Power on the VM through the Proxmox API.  |
| starting      | Verify running + email | Wait up to 5 cron ticks for the guest to report  running . Send the "VM is ready" email.                      |
| ready         | —                      | Final state. Client has full access to the service.   |

## Resumability & retry

Every step is wrapped in this contract:

1. **Check current state** — guards against double execution if the cron fires twice.
2. **Do one API call (or a short sequence of related calls)** — deliberately small so the step either completes quickly or can be retried cheaply.
3. **On success** — advance |vm\_status| to the next state.
4. **On failure** — return the error string; |vm\_status| stays the same; the cron log records the step with the error; next tick retries.

There is **no retry count limit**. A VM that genuinely cannot deploy (misconfigured IP pool, no free node with matching storage) will stay stuck at its current step and keep appearing in the cron log — visible immediately. Admins are expected to fix the root cause (add IPs to the pool, adjust storage mapping) rather than fight a phantom max-retry counter.

## DNS configuration during deploy

The |clone → set\_dns| transition is where forward and reverse DNS records are registered. With many IPs or many DNS zones, this can involve dozens of API calls. Starting with v3.2 each DNS operation is logged live to the cron output, so admins can watch records being created in real time:





# Migration step in detail

`set_dns → migrated` handles the common Proxmox cluster topology where templates live on one node (often for storage cost reasons) but client VMs should run on another:

1. `set_ip → clone` creates the VM on **Template Node A** using the template's local storage.
2. `clone → set_dns` configures DNS while the VM is still on Node A.
3. `set_dns → migrated` migrates the VM from Node A to the **Target Node B** with a `targetstorage` remap so disks end up on the right backend on the target.
4. All subsequent steps (CPU, disk, network, firewall, cloud-init, start) run against the VM on Node B.

Target node selection considers storage availability and free RAM on candidate nodes. If no suitable target is found, the VM stays on Node A and deployment completes there — the VM is still fully functional, just not on the preferred node.

## What triggers a deploy

A deploy starts the moment `vm_status` is set to `creation`. That happens when:

- A client's order is provisioned (automatic after payment, or manual accept/provision by an admin).
- An admin clicks **Create** or **Module Create** in the service's module commands.
- `ModuleCreateAccount` is called through the WHMCS API.
- A **Redeploy** action resets the VM record (deletes the existing VM on Proxmox, clears logs, sets status to `creation`).

From the moment `creation` is written, the next cron tick picks up the VM. At the default 1-minute **Process VMs** interval, that means provisioning starts within 60 seconds of the trigger.

## Related reading

- [Change Package](#) — how upgrades and downgrades use a similar state machine.
- [Terminate Process](#) — how services are torn down asynchronously.
- [DNS Zones & Integration](#) — configuring the providers that the `clone → set_dns` step writes to.
- [Scheduled Tasks](#) — all cron tasks including Process VMs.

---

Revision #8

Created 10 April 2026 19:08:14 by Ruslan

Updated 15 May 2026 14:07:35 by Ruslan