Deployment & Segmentation

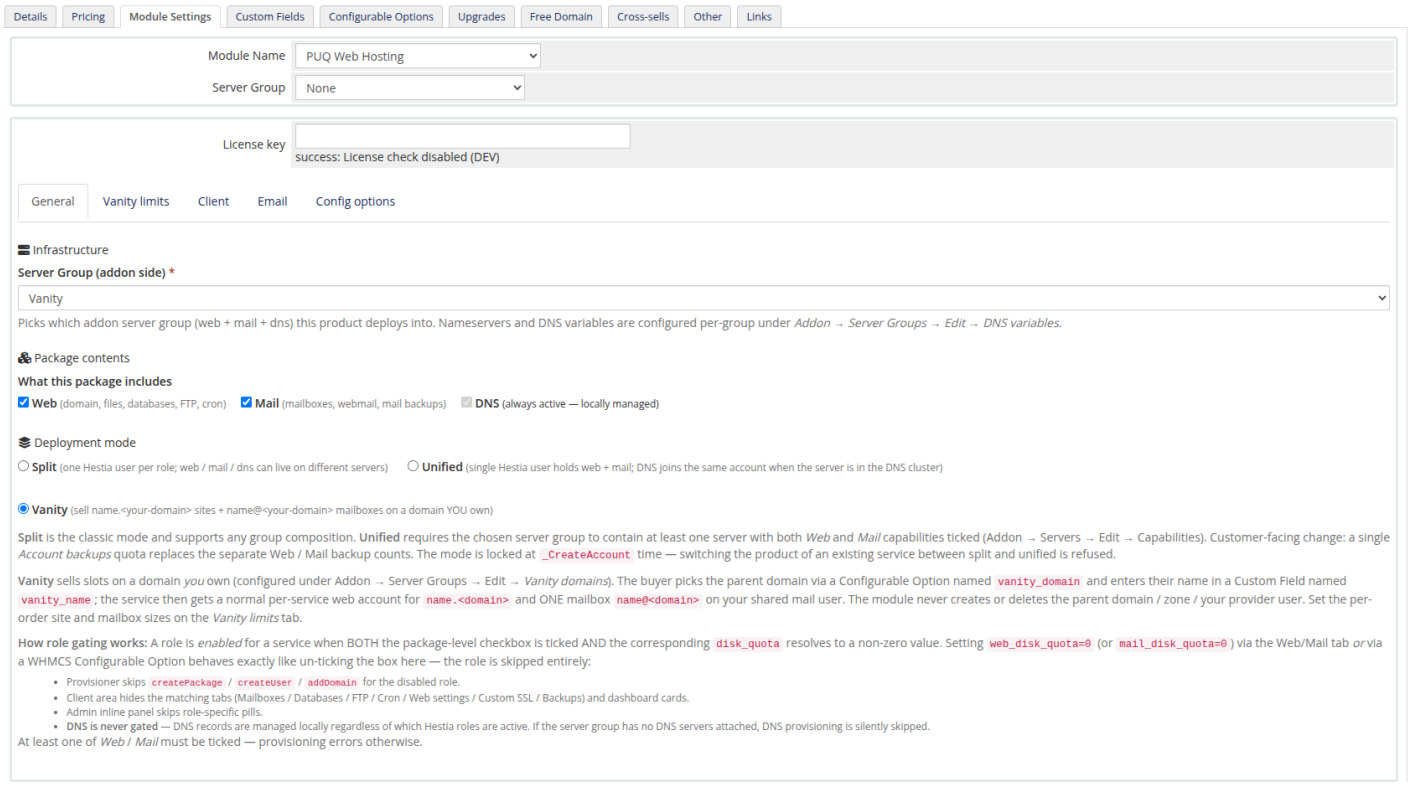
This chapter explains the heart of the module: how a service is deployed, why you split your infrastructure across servers, and the three deployment models (Split, Unified, Vanity) you can sell. Read it before building products — the choices here decide how your hosting scales and how your costs and isolation trade off.

Deployment Models (Split · Unified · Vanity)

PUQ Web Hosting module **WHMCS**

Order now | Download | Community

Every product you sell has a **Deployment mode** set on the product's *Module Settings → General* tab. It decides **how many HestiaCP accounts** back each service and **where** the web, mail and DNS roles live. This is the single most important product decision.



The three modes

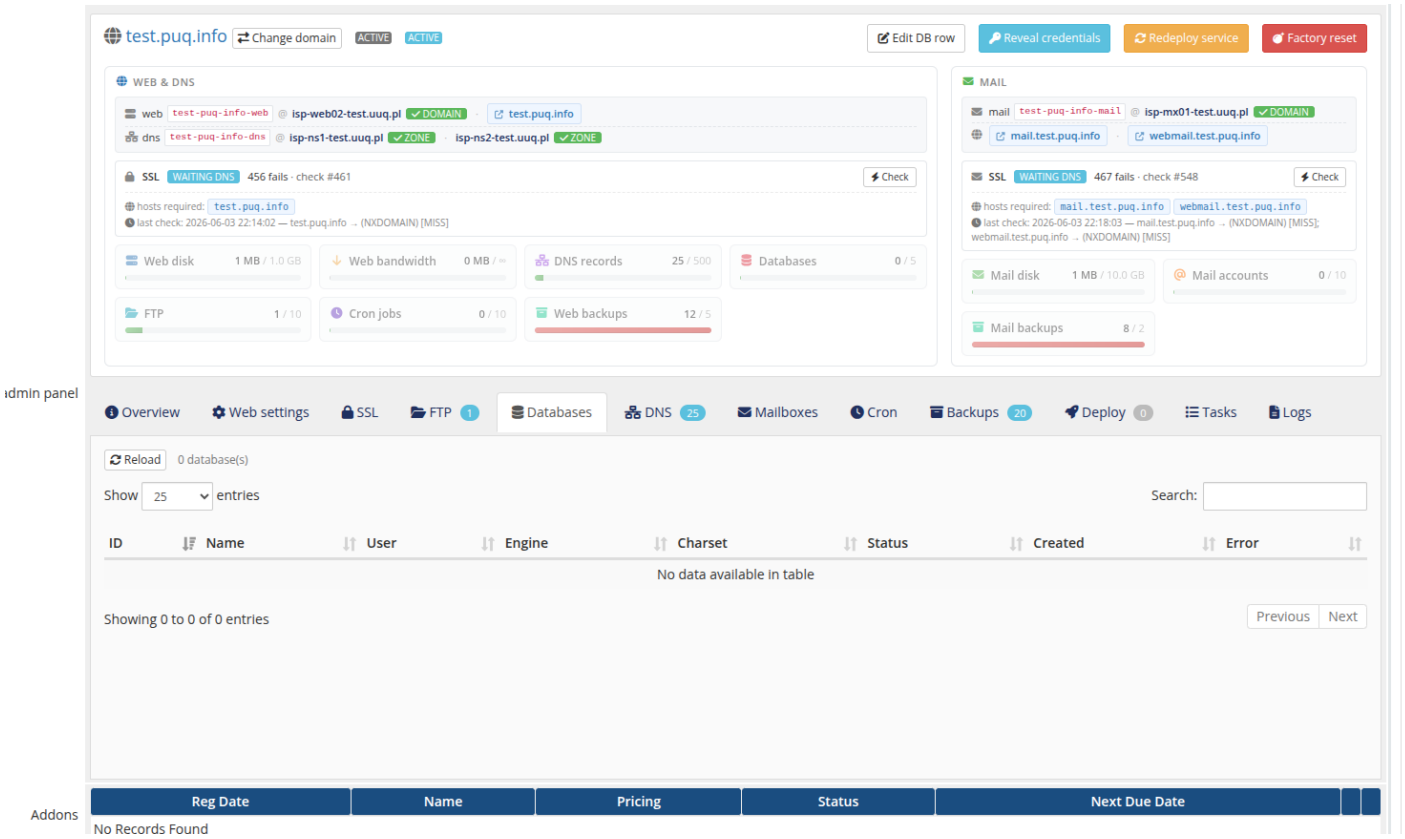
Mode	Hestia accounts per service	Best for

Split	A **separate Hestia user per role** — one web user, one mail user, oneDNS user (each can live on a different server).	Classic shared/business hostingwhere you want maximum isolationand the freedom to put web, mail andDNS on different,independently-scaled servers.
Unified	**One Hestia user** holds web **and** mail (and local DNS) for the service.	Simpler, denser, cheaper hosting where web and mail naturally live together on one node.
Vanity	A **per-service web user** for one subdomain + **one mailbox** on your **shared provider** mail account.	Selling `name.yourdomain.com` websites and `name@yourdomain.com` mailboxes on a domain *you* own — see the dedicated **Vanity Mode** chapter.

Split deployment

In Split mode the module provisions up to three independent HestiaCP users for the service — for example `customer-com-web`, `customer-com-mail` and `customer-com-dns`. Each is placed on a server that has the matching capability, so the **website**, the **mailboxes** and the **DNS zone** can sit on completely different machines.

The admin service panel shows this clearly — a **Web & DNS** card and a separate **Mail** card, each with its own Hestia user, server and certificate:



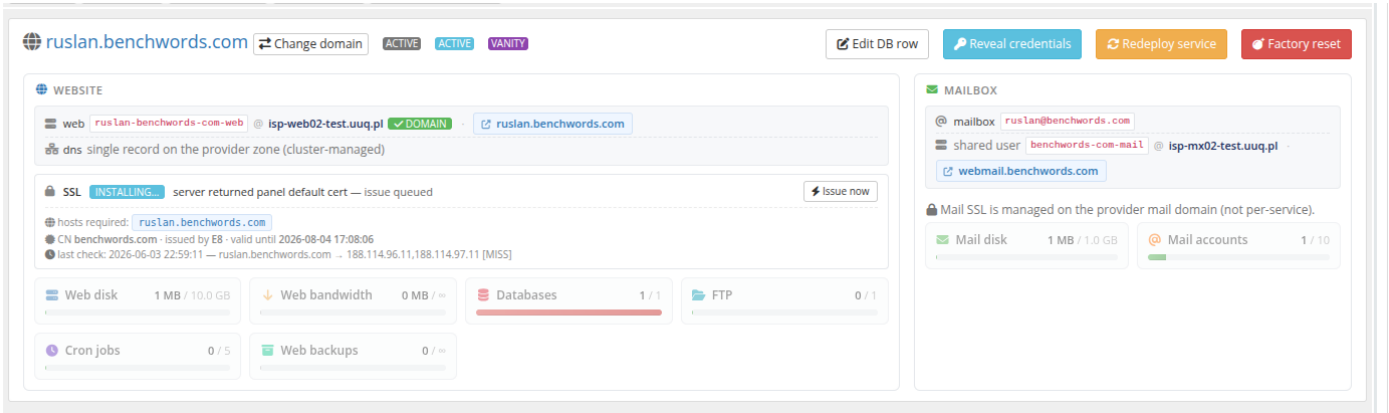
Split is the mode to choose when you want to **segment your fleet by role** (the next page) and keep mailboxes off your web servers.

Unified deployment

In Unified mode a single Hestia user owns the website and the mailboxes, so web and mail are always co-located. There is no separate mail user to manage. Unified products use a single combined account package and a single set of backups (`role = all`). Choose Unified when you run general-purpose nodes that do both jobs and you want fewer accounts to manage.

Vanity deployment

Vanity is a different business model rather than a different server layout: you own a domain (e.g. `benchwords.com`) and sell **slots** on it. Each order becomes `name.benchwords.com` (a normal per-service web account) plus `name@benchwords.com` (one mailbox on your **shared** provider mail user). The parent domain, its DNS zone and the provider mail account are **never** modified per order — the model is destructive-safe by design.



Vanity has its own chapter because the setup (sellable parent domains, reserved names, the order flow and a drop-anywhere shop widget) is substantial — see **Vanity Mode**.

Which roles does a product include?

Independently of the mode, the product's *General* tab lets you tick which **roles** the package includes — **Web**, **Mail**, **DNS**. Unticking a role (or setting its disk quota to `0` in the limits tab) simply omits it. For example, a "mail-only" product ticks Mail and leaves Web off; a "website-only" product ticks Web and DNS.

The limits for each ticked role are configured on the matching *Web limits* / *Mail limits* / *DNS limits* tab (or the single *Vanity limits* tab in Vanity mode) — see **Installation & Configuration → Create a product**.

> **Rule of thumb:** Split = isolation + segmentation; Unified = density + simplicity; Vanity = a productised "personal site + email" offer on your own domain. You can sell all three side-by-side — they coexist on the same servers.

Server Segmentation (Web / Mail / DNS tiers)

PUQ Web Hosting module **WHMCS**

Order now | Download | Community

One server or many?

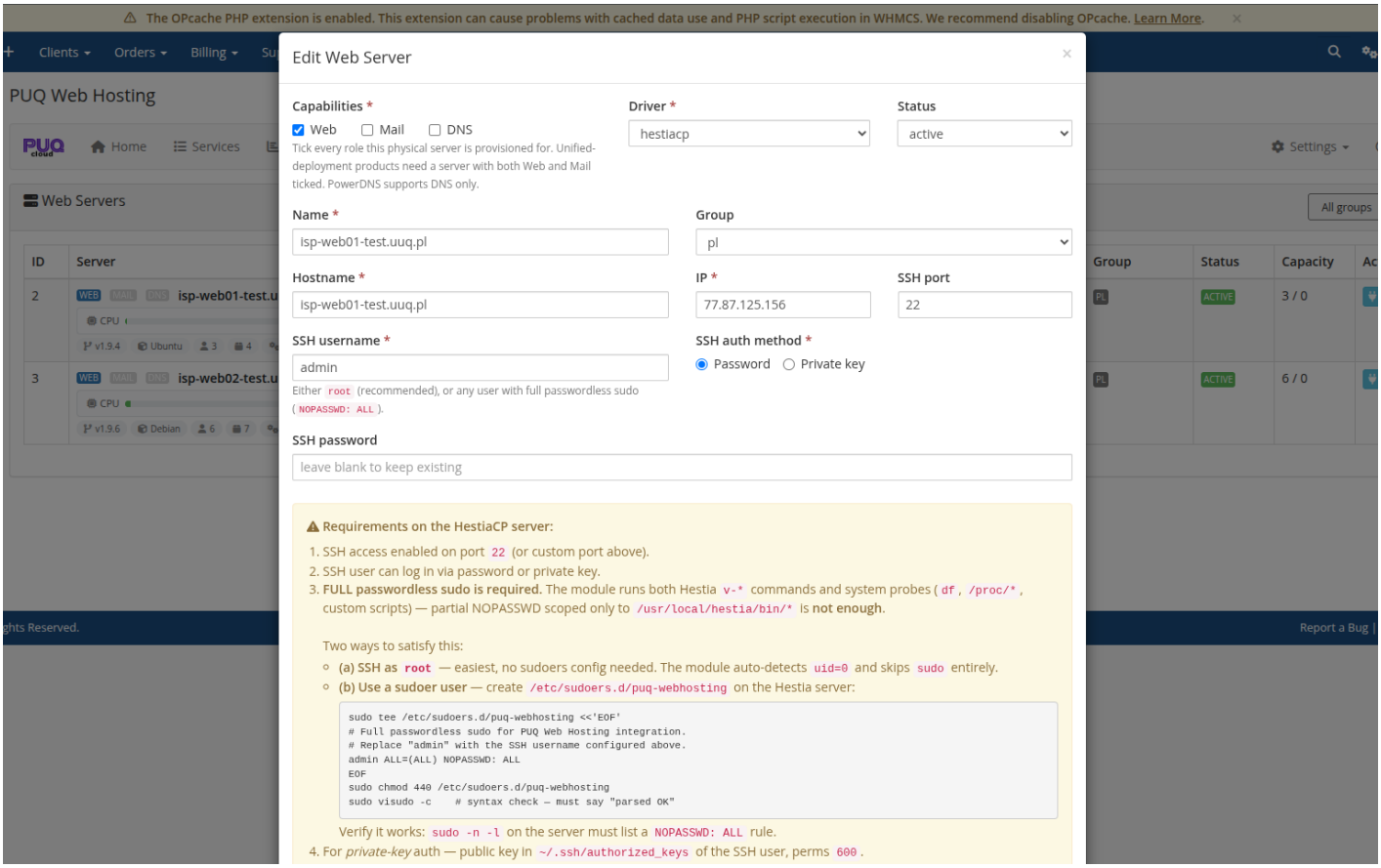
You can run the whole module on a **single** HestiaCP node that does everything. That is perfect for getting started, for small offers, and for testing.

But the module is designed so you can **segment** your fleet as you grow: put **websites** on one set of servers, **mailboxes** on dedicated **mail (MX)** servers, and **DNS** on separate **nameservers**. Each role then scales on its own hardware, and a busy mail queue or a heavy website can't starve the others.

The mechanism that makes this possible is **server capabilities** + **server groups**.

Capabilities — what a node is allowed to host

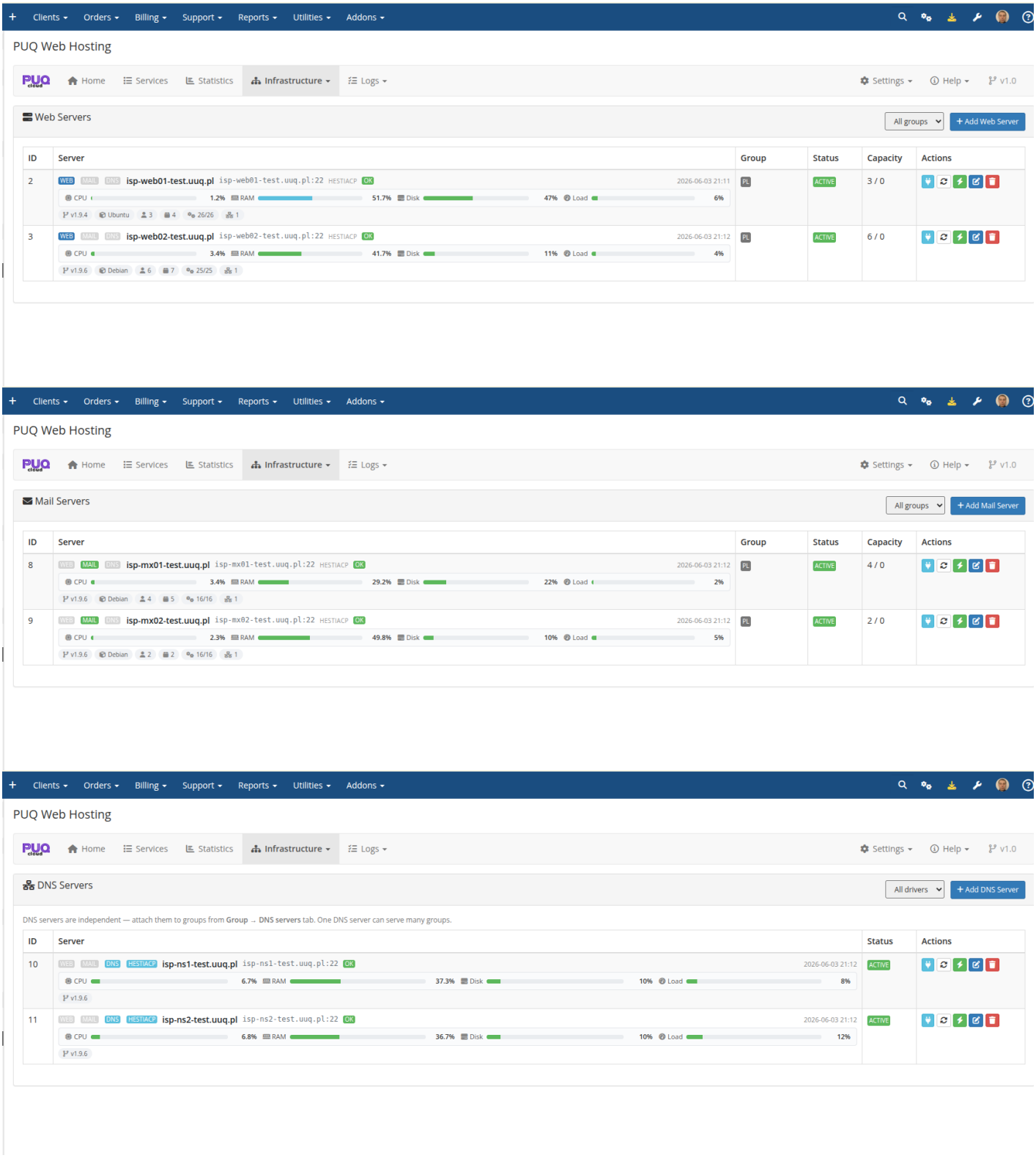
Each server you add (Infrastructure → *Web / Mail / DNS Servers*) has three capability flags: **Web**, **Mail**, **DNS**. You tick them when you add or edit the node. A node can carry one role, two, or all three.



This is how you decide your topology. Typical patterns:

Pattern	Web nodes	Mail nodes	DNS nodes
All-in-one	one node ticks Web + Mail + DNS	(same node)	(same node)
Web / mail split	nodes tick **Web** only	dedicated **Mail** nodes (your MX)	nodes tick **DNS** only
Full three-tier	Web pool	Mail pool	Nameserver pool (e.g. `ns1`, `ns2`)

In the Infrastructure pages the same physical servers appear under **Web Servers**, **Mail Servers** and **DNS Servers** filtered by their capability — so a mail-only node only shows up under Mail:

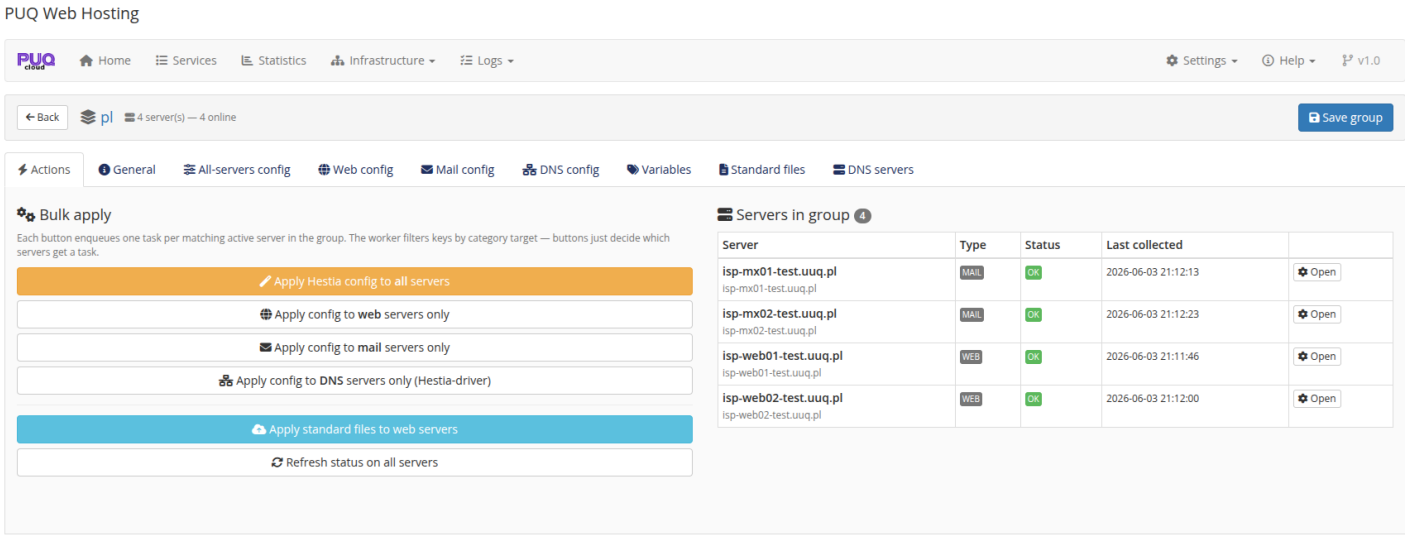


When a **Split** service is provisioned, the module places each role on a **least-loaded active node that has the matching capability**: the web user lands on a Web node, the mailbox on a Mail node, the DNS zone on the group's DNS nodes. That is the load balancing — it is **by role and by capacity**, automatically.

Server groups — the unit you sell

from

A **server group** ties a set of nodes together and is what a product points at (the product's *Server Group* dropdown). Open a group to manage it.

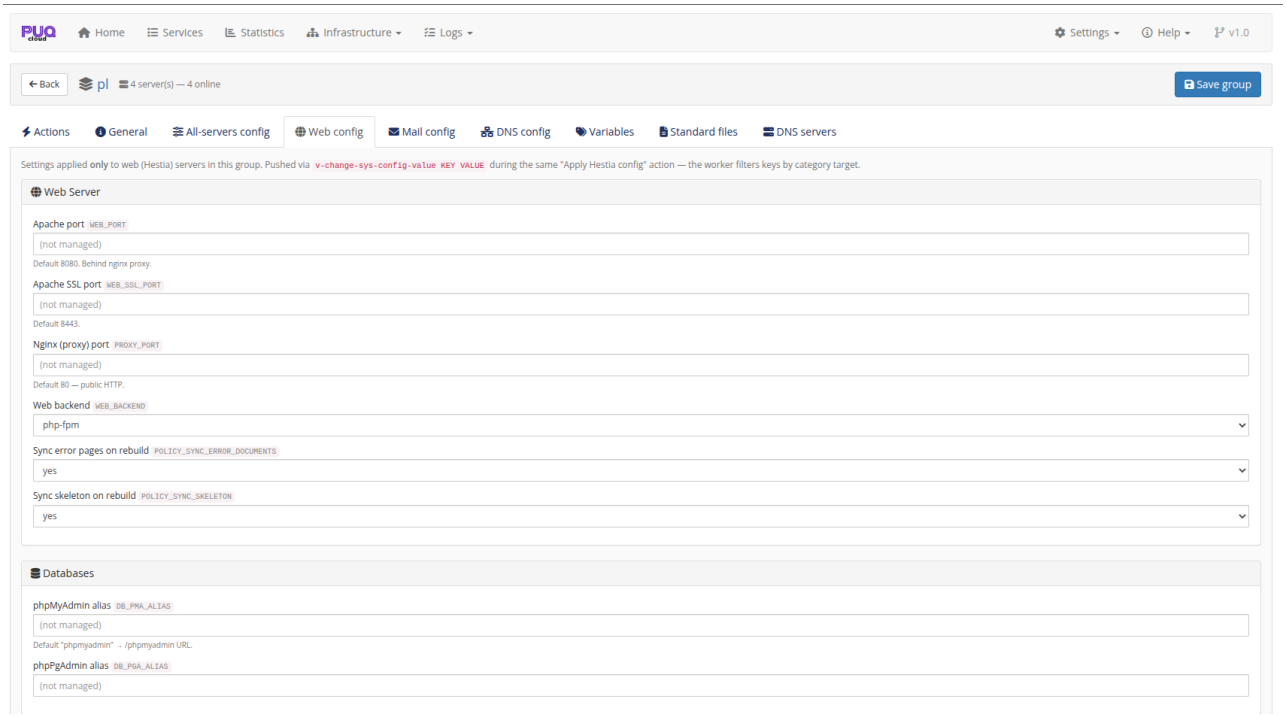


The group's **Actions** tab shows the members with their **Type** (WEB / MAIL) and lets you push configuration to **all servers**, **web servers only**, **mail servers only** or **DNS servers only**. That last point is the key to safe segmentation: a web-specific setting is **never** applied to a mail node and vice-versa.

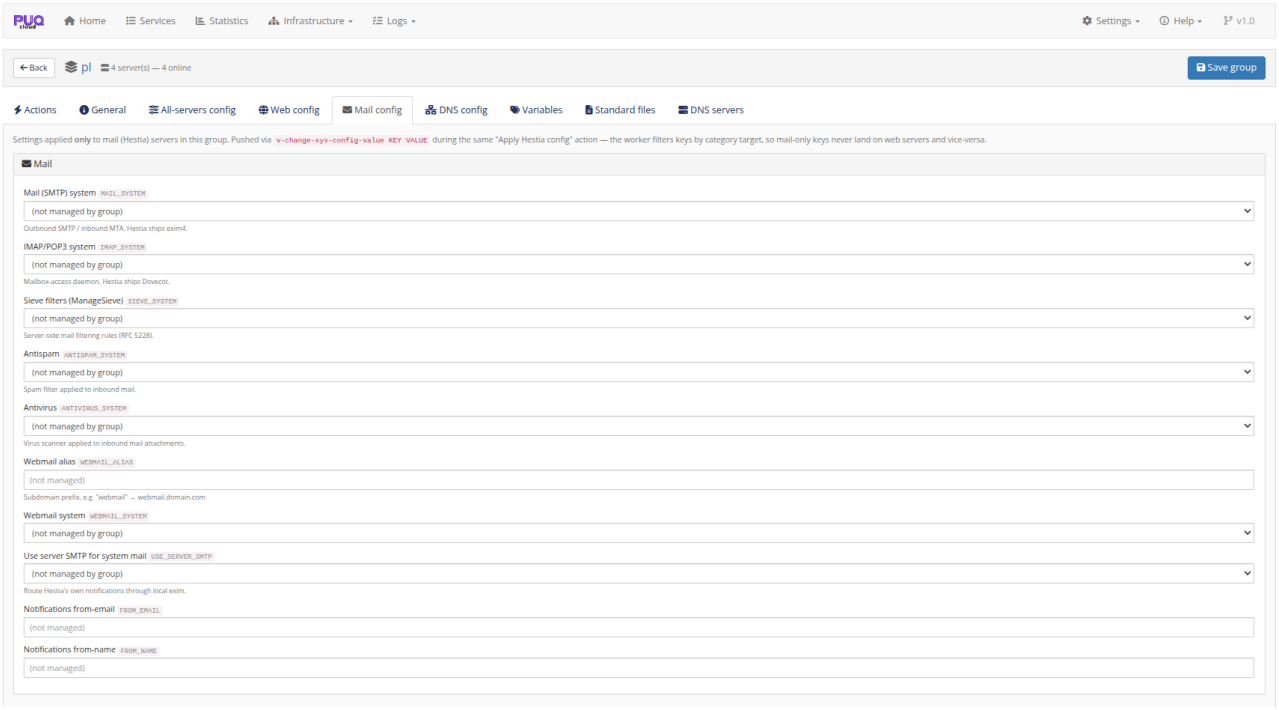
Role-targeted configuration

The group editor has separate config tabs whose values are pushed **only** to nodes of that role during *Apply Hestia config*:

- **All-servers config** — keys applied to every node (language, theme, security/API, user policies…).
- **Web config** — web-only keys (Apache/Nginx ports, web backend, phpMyAdmin alias…).



- **Mail config** — mail-only keys (SMTP/IMAP system, antispam/antivirus, webmail, notification from-address…).

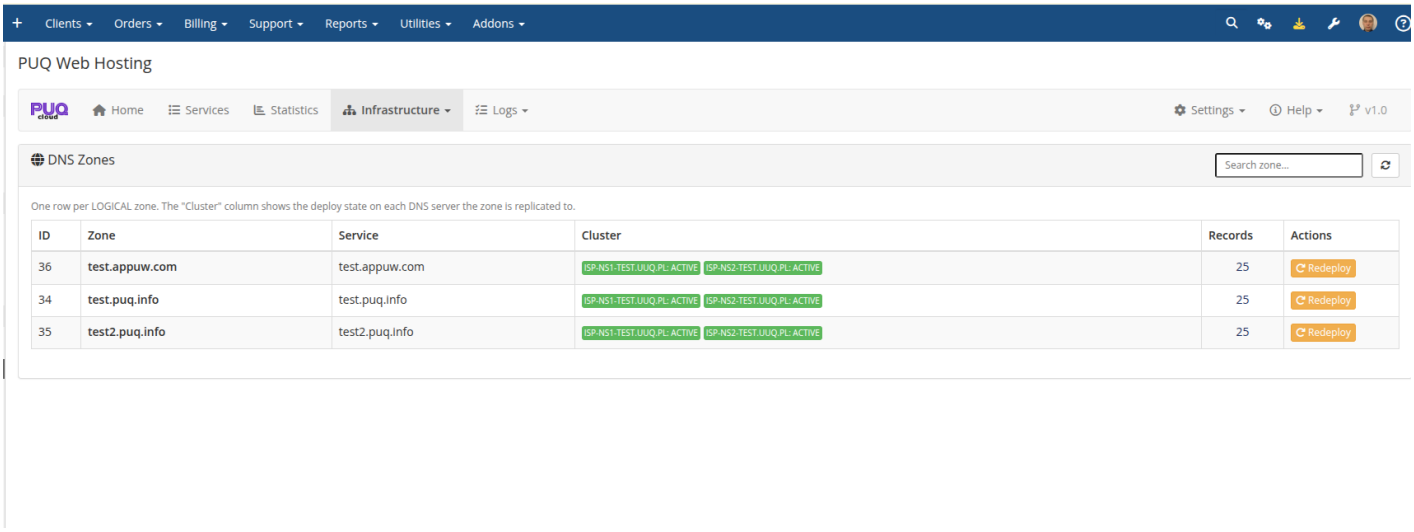


- **DNS config** — DNS-only keys (DNSSEC, DNS cluster). PowerDNS nodes have no `hestia.conf` and are skipped automatically.

The worker filters every managed key by its category target, so "mail-only keys never land on web servers and vice-versa". This means you can, for instance, enable ClamAV/SpamAssassin only on your mail tier without touching the web tier.

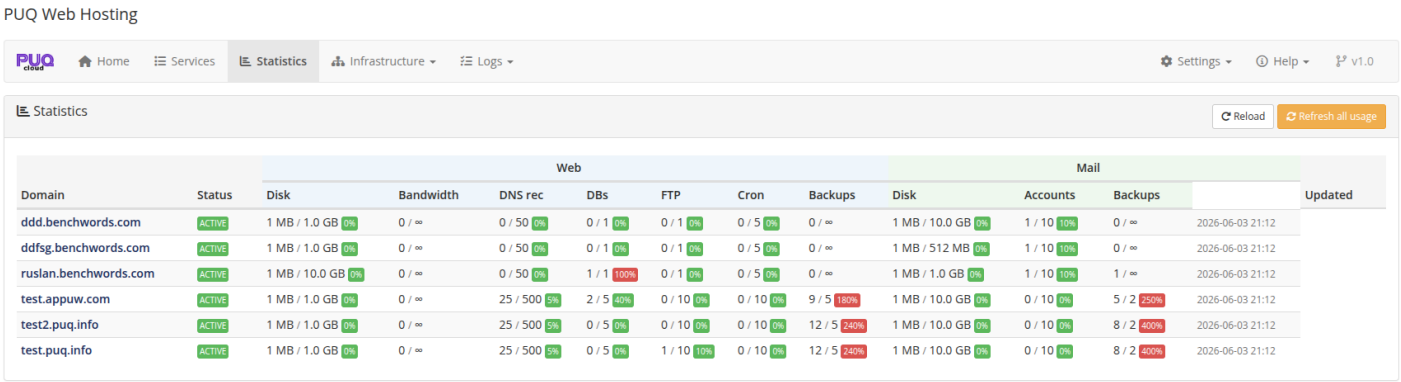
DNS is active-active across the tier

DNS servers are **independent** — you attach them to a group on the group's **DNS servers** tab, and **one DNS server can serve many groups**. When a zone is created it is **replicated to every attached DNS node**, so you get an active-active nameserver cluster out of the box. The **DNS Zones** page shows the per-server deploy state for each logical zone:



Watching the balance

The addon **Statistics** page reports usage **per domain, split into Web and Mail columns**, so you can see at a glance how load is distributed and when a tier needs another node:



The Home dashboard's **Server capacity** table lists each node by type with its capacity and last sync time — a quick health view of every tier.

> **Why segment?** Mail and web have very different load profiles. Keeping them on separate, capability-tagged tiers lets you scale each independently, isolate failures, and apply role-specific tuning — while the module's per-capability

> placement spreads new accounts across the least-loaded node automatically.

How Deployment Works

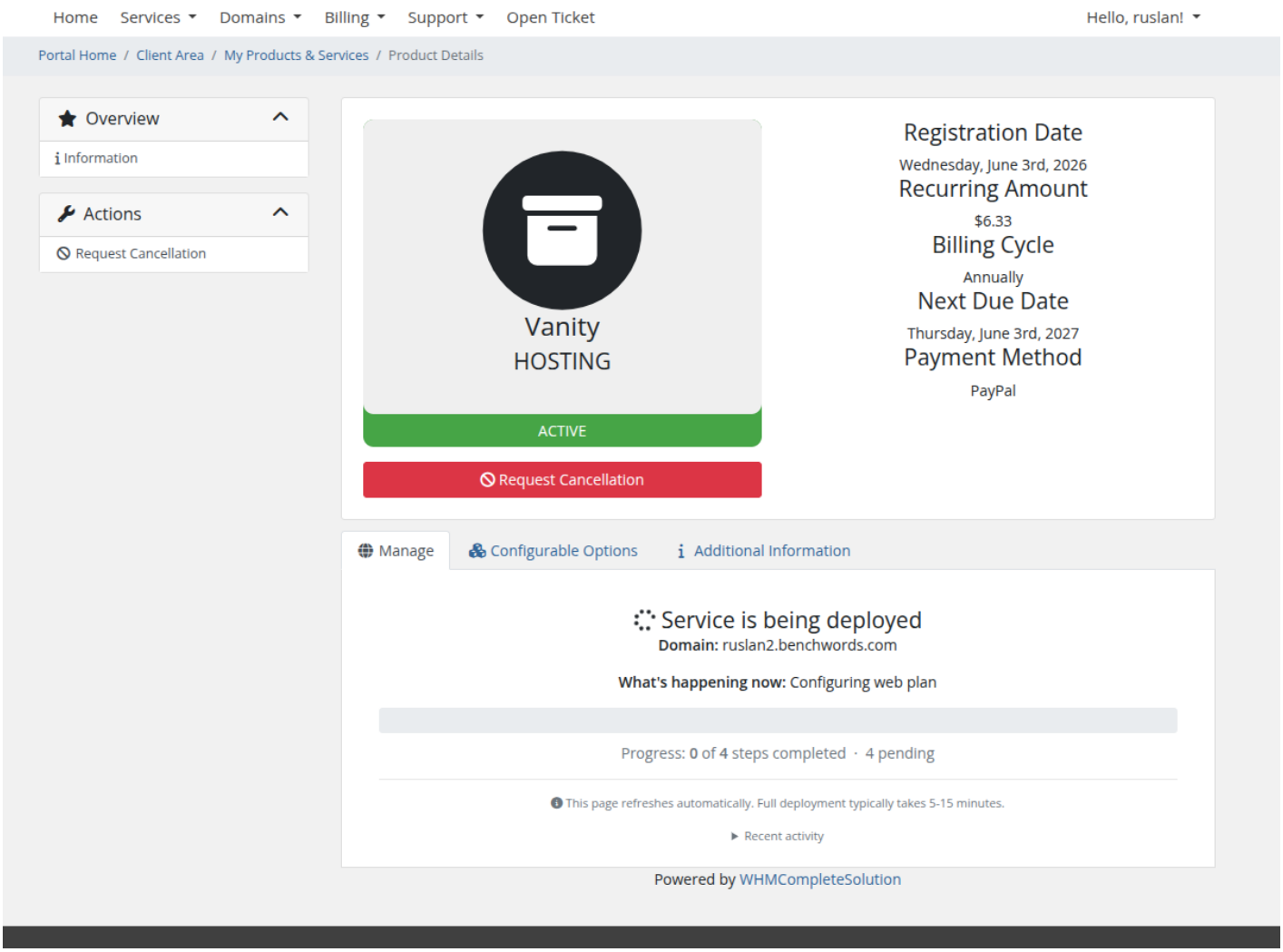
PUQ Web Hosting module **WHMCS**

Order now | Download | Community

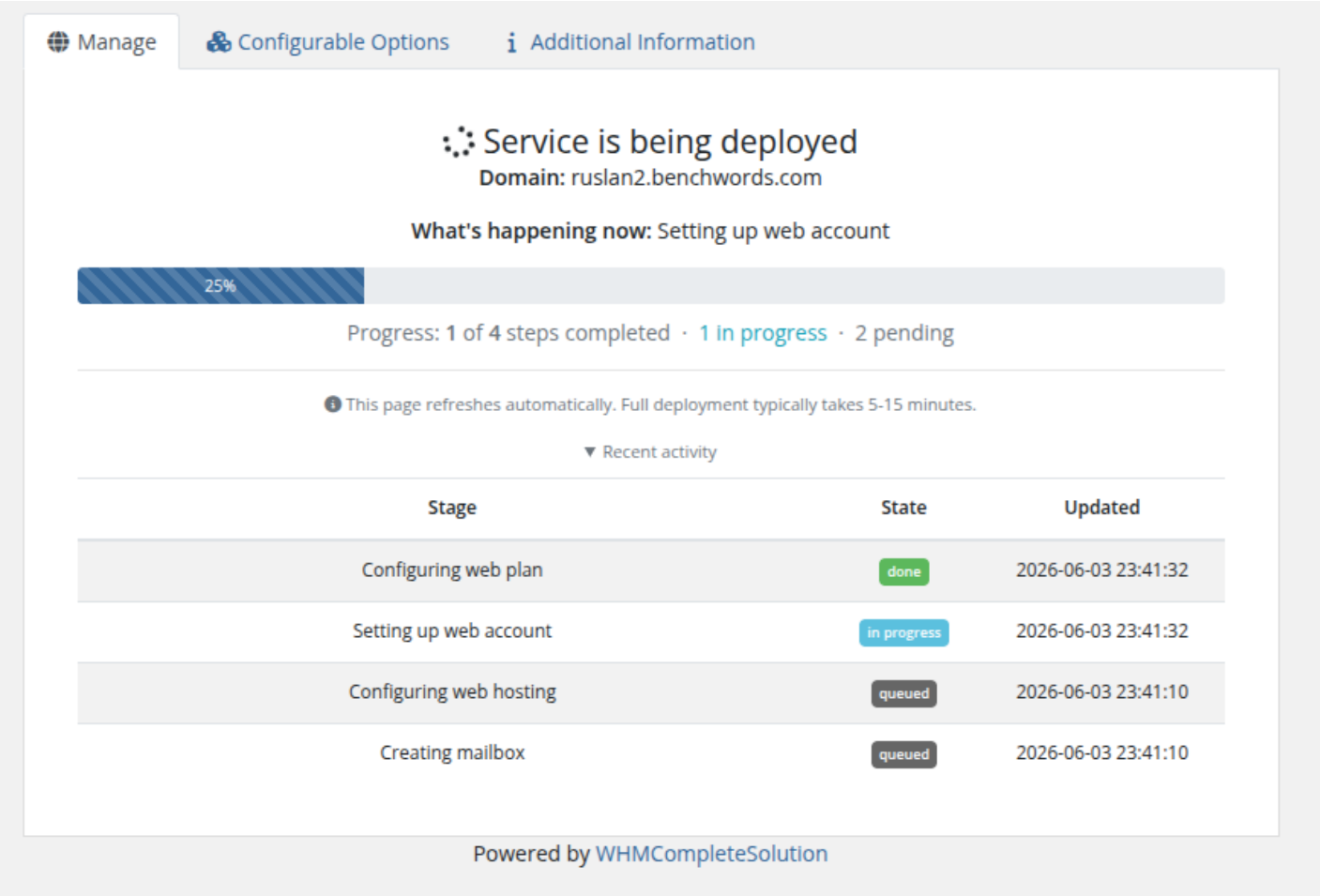
Orders return instantly — work happens in the background

When a service is created (a WHMCS order goes active, an admin clicks **Create**, or the API is called) the module does **not** try to build everything inside the HTTP request. It writes a service record, returns `success` to WHMCS immediately, and lets the **cron-driven task queue** do the real work.

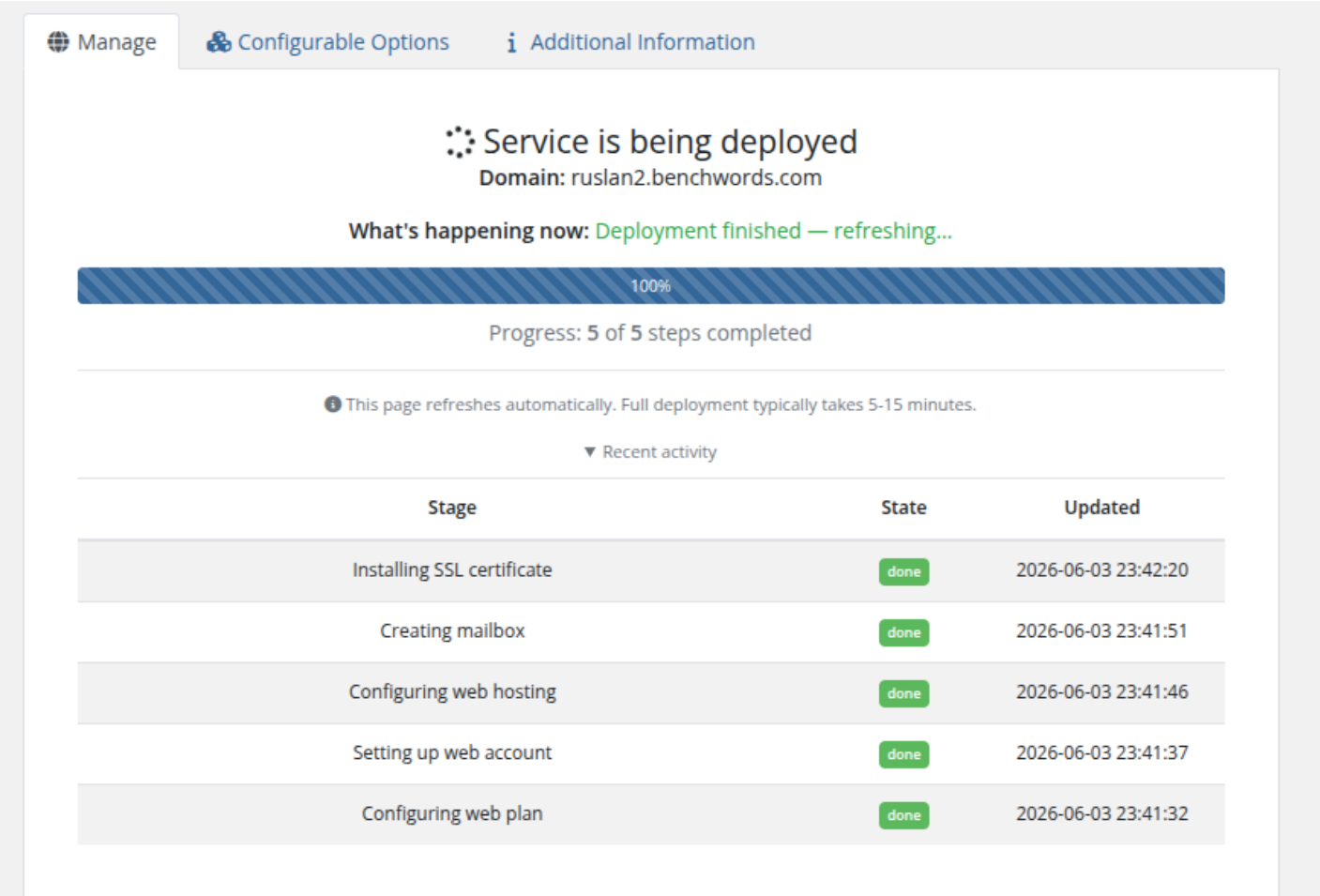
This is what the customer sees right after ordering — a live deployment splash that polls for progress and refreshes itself:



Each step is a queued **task** processed by the cron runner. As steps complete, the bar advances and the recent-activity feed updates:



When the last step finishes the page flips to the live control panel:



This design makes provisioning resilient to slow panels, SSH timeouts, apt-get locks and long operations — none of them block the order, and any failed step is simply retried.

The pipeline (Split example)

A Split service is built by a chain of idempotent tasks, roughly in this order:

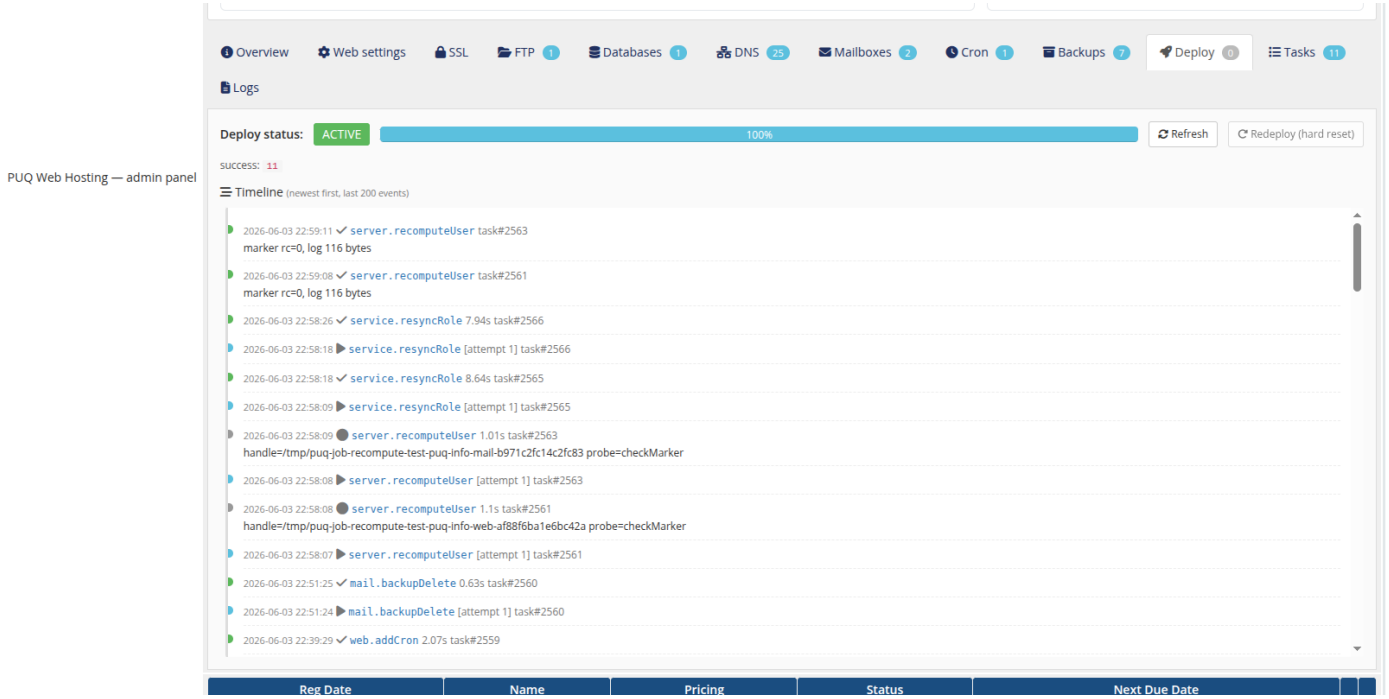
1. **Create the web package + web user** on a Web node.
2. **Add the web domain** (and the web account's quota/PHP settings).
3. **Create the mail user + mail domain** on a Mail node (with DKIM).
4. **Create the DNS user + zone** and **deploy the zone** to every attached DNS node.
5. **Seed Auto-SSL** so Let's Encrypt issues as soon as DNS resolves.

Unified runs the same idea with one combined account; Vanity runs a trimmed chain (one web account, one mailbox on the shared mail user, one DNS record). Every task is **idempotent** — re-running it is safe — so the chain can resume from wherever it stopped.

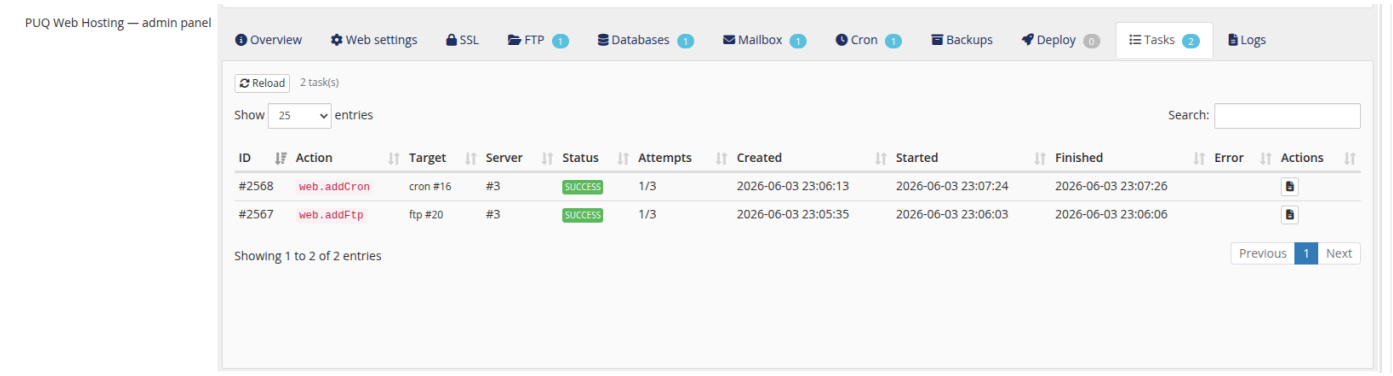
Watching and driving it from the

admin panel

The admin service panel's **Deploy** tab shows the live deploy status and a full **timeline** of every task with its duration and result, plus a **Redeploy (hard reset)** button:



The **Tasks** tab lists this service's tasks (action, server, attempts, timestamps); each row opens a detail modal with the streamed SSH log:



Retries, failures and tickets

A failed task is retried with exponential back-off. Once all attempts are exhausted it is **terminally failed**, the service shows a deploy error, and — if you enabled it — the module can open a **support ticket** for your staff. The retry policy and the failure-ticket behaviour are configured in addon **Settings** (see *Cron & Automation → Task queue, retries & tickets*).

> **Key takeaway:** deployment is an asynchronous, idempotent, resumable pipeline. The order is accepted instantly, the queue does the work, failures self-heal on retry, and both you and the customer get live progress.