High Availability with Proxy Groups in Zabbix

HA from server to the edge — what changed with 7.0

Christian Anton — Inqbeo — Zabbix Conference Germany 2026

ZABBIX '26 CONFERENCE GERMANY

Inqbeo

IT Automation & Reliability

ZABBIX

A specialized IT consulting company helping customers adopt modern open-source and cloud-native technologies — with a focus on the two areas at the heart of this talk.

Monitoring & Observability

Zabbix • Prometheus • Thanos • VictoriaMetrics • Grafana • OpenTelemetry • Loki

Containers & Automation

Kubernetes • RKE2 / K3s • Rancher • Harvester • GitOps (Flux / ArgoCD) • Ansible

Architecture Consulting

Solution Development

Operations & Support

Trainings & Workshops

inqbeo.de • **Christian Anton** Owner | Consultant | Trainer

Why HA in monitoring at all

ZABBIX

Monitoring is the senses — if it goes dark, the rest goes dark with it

01

Flying blind

Monitoring is the senses of the platform. If it is down, you lose your senses.

02

Longer MTTR

Blind spots during outages stretch resolution time, breach SLAs, trigger compliance findings.

03

Your data is your reputation

Zabbix is a great reporting tool. Customer-facing dashboards and SLA reports with gaps in the graphs damage trust — and invite questions you don't want.

04

Automation stops

Self-healing infrastructure depends on monitoring signals — no signal, no trigger, no automation.

HA primer: stateless vs. stateful

A frame of reference — not a distributed-systems crash course

⚠ **Heads up:** ***HA is genuinely hard.*** No silver bullets — every layer adds operational complexity, every layer fails in surprising ways.

Stateless

Data flows ***through*** — no instance holds anything important.

N identical instances behind a load balancer.

- Trivially horizontally scalable
- Cheap to operate, easy to reason about

Stateful

Instances ***hold*** data — now you own a distributed-systems problem.

Requirements

- Data replication (sync vs. async)
- Failure detection (heartbeat, lease, gossip)
- Leader election / quorum (often 2N+1)
- Client-side failover (VIP, DNS, conn-string lists)

Workaround: externalize state

Push state to a central HA store — **DB cluster, S3, Redis** — and the service becomes “stateless” again.

But: state is also data. *Processing often can't be evenly spread — the Zabbix Server is exactly such a case.*

Two different HA models in Zabbix

Server/DB HA is classic active/standby. Proxy Groups are something else.

Zabbix Server & DB

Classic active / standby HA

Two stateful services with shared coordination state.

- **DB cluster** — Patroni / Galera / CNPG; the foundation.
- **Server pair** — active/standby since 6.0; coordinates via the DB.
- One node serves at a time — standby is idle capacity, not extra throughput.

⚠ **Genuinely complex.** *Out of scope today — covered briefly on the next slide.*

Zabbix Proxies (Proxy Groups)

"Cattle" + server-managed load-sharing

Not active/standby. Every proxy in the group is active and processing hosts.

- **Cattle, not pets** — identical, interchangeable, no per-instance state to babysit.
- **Server is the controller** — it owns host→proxy assignment, fairness, and failover.
- **Load-sharing** comes built-in — often more valuable than the HA itself.

→ Proxy Groups make sense **even without server HA**.
They're solving collection scale & resilience, not server uptime.

Two different problems, two different solutions. *This talk is about the right one.*

Zabbix Server & DB HA — briefly

One slide of context — Proxy Groups don't depend on either

Not the focus of this talk — *but worth a one-slide reference. Proxy Groups work independently of either.*

Server HA

Active / standby pair, since 6.0

The DB is the coordination plane — no Pacemaker, no Keepalived.

Failover

~5 s graceful • ~1 min default
(tunable 10 s – 15 min)

Config per node

```
HANodeName= + NodeAddress=
```

⚠ Not load balancing.

Standby is idle capacity, not extra throughput.

DB HA

The actual foundation

If the DB goes, Zabbix goes. Independent of Server HA.

PostgreSQL

Patroni (+ etcd) • CNPG • Zalando

MySQL / MariaDB

Galera • Percona XtraDB • Group Replication

⚠ Replication != HA.

Replication only solves data motion. You still need leader election, stable client endpoint, automatic failover.

Pragmatic picks

Greenfield on Kubernetes

CNPG (PostgreSQL) — default
Percona Operator (MySQL)

Brownfield VM estate

Patroni or Galera — use what your ops team knows

Validate

Failover is automatic
Endpoint is stable (VIP / PgBouncer / ProxySQL)
Backups + PITR — HA is not a backup

Why use proxies — and what they cost you

Proxies solve a lot. They also bring their own problems.

WHY

Proxies are a scaling & resiliency primitive

- ✓ **Offload collection**
  Server focuses on triggers, escalation, history/trends.
- ✓ **Local buffering**
  During server outages, upgrades, failover windows.
- ✓ **Near-target collection**
  Latency, reliability, local DNS/domain handling.
- ✓ **Fewer firewall pinholes**
  One opening to the proxy instead of N×M.
- ✓ **Passive → active cost shift**
  SNMP/HTTP polling cost on proxy. Server just ingests.
- ✓ **Safer ops**
  Upgrades, migrations, Server HA failovers less scary.

BUT

They come with their own operational reality

- ⚠ **They are stateful**
  Each proxy carries a config cache and a collected-data buffer (SQLite / MySQL / Postgres). Not just another web tier.
- ⚠ **Endpoints have to talk to them**
  Agents need ServerActive=, traps need a target, network devices need an SNMP manager IP. Misroute and data is lost.
- ⚠ **Often at the edge**
  Remote sites, DMZs, customer networks. Each becomes a single-proxy bottleneck — and a long way from your ops team.
- ⚠ **More operational surface**
  More boxes to patch, monitor, version-pin. Config parity (scripts, ODBC, creds, TLS) becomes a discipline.

So: how do you make ***these*** — stateful, edge-located, endpoint-coupled — highly available?

Making proxies HA: the old way

Pre-7.0 — partly solvable, but never clean

We've now seen that proxies are essential and **the rest of the Zabbix stack can be made highly available** — server, database, the lot. ***So what about the proxies themselves?***

The need has always been there. Until 7.0 it was never really addressed by Zabbix — so operators built their own. **Proxies are stateful** (config cache + collected-data buffer in SQLite / MySQL / Postgres / in-memory hybrid), which makes "just run two" harder than it looks.

⚠ PARTLY WORKS

VIP + keepalived / Pacemaker + shared storage

You CAN run a proxy active/passive with classic Linux-HA tooling: keepalived or Pacemaker float the VIP, shared storage (NFS, DRBD, SAN) holds the proxy's local DB and buffer. The standby node takes over on failover.

But: heavy. Resource agents, fencing/STONITH, split-brain, shared storage (itself a SPOF you pretend is HA), version drift, careful Zabbix runtime-config handling. Most teams get it 80% right and live with surprises.

⚠ PARTLY WORKS

Trigger → Ansible / AWX reassignment

A trigger fires when a proxy goes offline. An action runs an Ansible / AWX playbook that talks to the Zabbix API and re-points hosts to a healthy proxy.

But: agents must be reconfigured too (otherwise active agents flood themselves AND the proxy with errors trying every entry in ServerActive). Playbooks aren't fast. Real operational effort to keep the playbooks, inventory, and credentials current.

× TRIED, FAILED

Pairs + manual API reassignment

Two identical proxies, a hand-written script that flips host assignments via the Zabbix API when one dies. Toil, race conditions, audit-log noise. Every script author thinks theirs is finally the right one — it isn't.

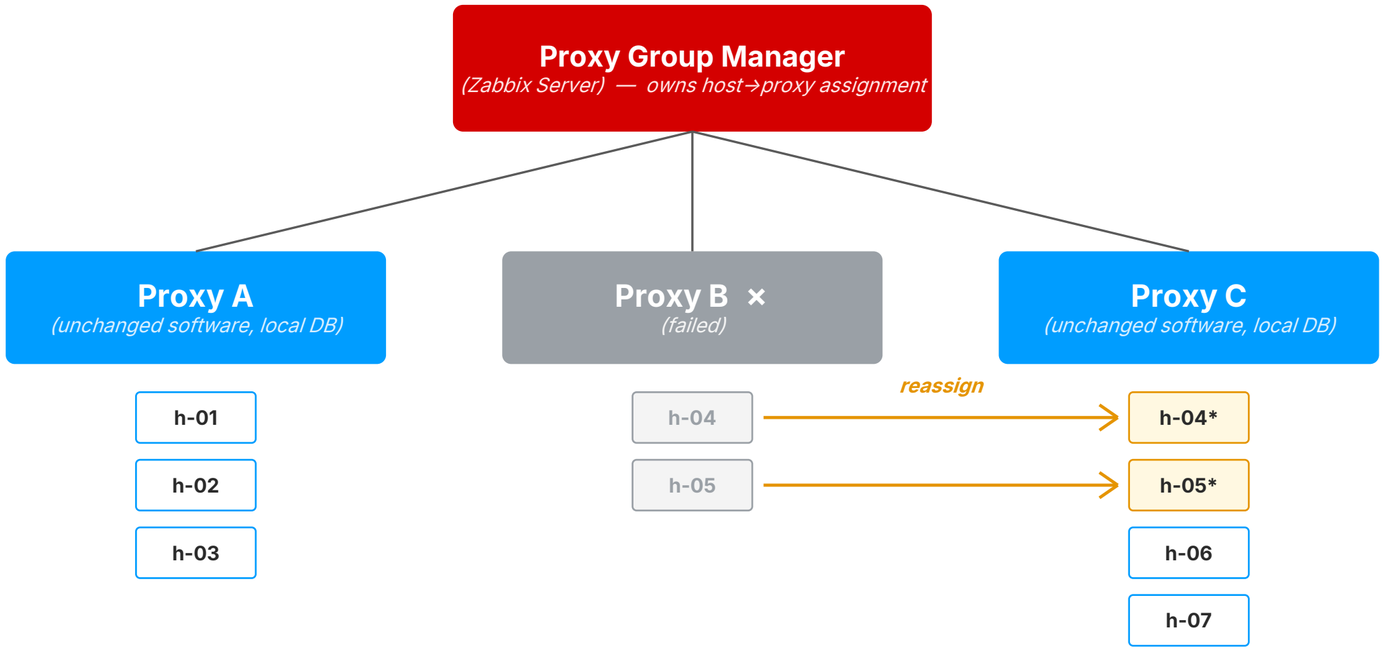
Either way: the server had no awareness of "proxy A is a replacement for proxy B."
→ *Most environments ran single proxies and accepted the outage window.*

Enter Proxy Groups

Smart server, dumb proxies — minimal change to the proxy itself

Since Zabbix 7.0 LTS

A logical set of proxies the server treats as one collection



Smart server, dumb proxies.

The proxy software **didn't change**.

A grouped proxy doesn't need to know it's part of a group. It just collects what the server asks for.

- → No new clustering protocol
- → No proxy-to-proxy chatter
- → No shared storage

Architecture stays simple. Value goes up.

High Availability

Proxy failure → instant host reassignment. No edge clustering, no VIPs, no shared storage.

Load Balancing

Even host distribution across the group. Often more valuable than HA itself.

Proxy Groups: how the server decides

Two knobs. Two fairness rules. One grace period.

Failover period

1 min default (10 s – 15 min)

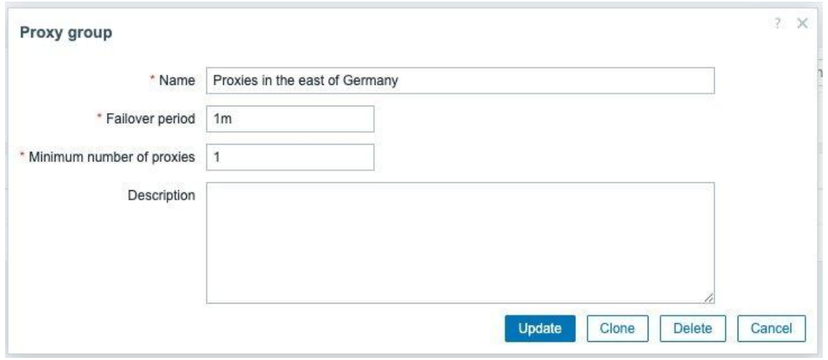
Silence window before a proxy is declared offline. Also the heartbeat expectation.

Min online proxies

1 default (1 – 1000, macros OK)

Fall below this and the group itself goes offline.

From the Zabbix UI — *Administration → Proxy groups*



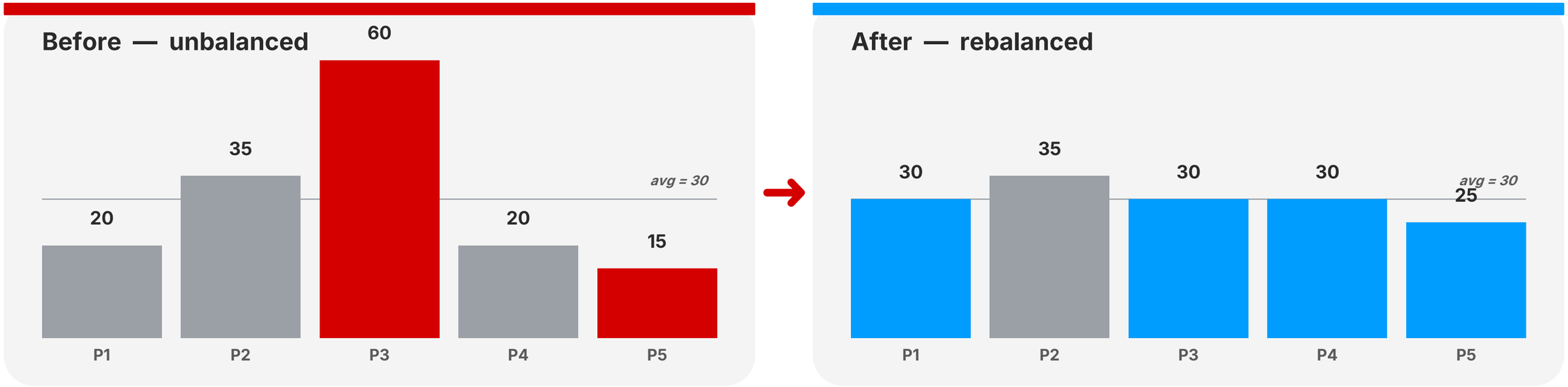
Host-to-proxy assignment

- Host is assigned to the **group**, not an individual proxy
- Initial placement: **proxy with the fewest hosts**
- Failover: **immediate** on proxy-offline detection — orphans go to least-loaded survivors
- Continuous rebalance: triggered only on **host-count imbalance** *(details on the next slide)*

⚠ **Small groups (< 10 hosts) will look uneven.** *Documented. Expected. Not a bug.*

Rebalancing in practice

Worked example — 5 proxies, 150 hosts



1 DETECT
Both conditions must hold, simultaneously

|count – avg| ≥ 10 hosts (the absolute difference)
AND count ≥ 2× avg OR count ≤ ½× avg (a factor of 2 above or below)

2 WAIT
Grace period before any move

10 × failover_period must persist throughout
Default failover = 1 min → ~10 minutes wait. Set 10 s and it's ~100 s.

3 REDISTRIBUTE
Move just enough — not necessarily perfect balance

Excess proxies unassign down to avg → unassigned pool
Pool drains to least-loaded proxies first (deficit may exceed pool size)

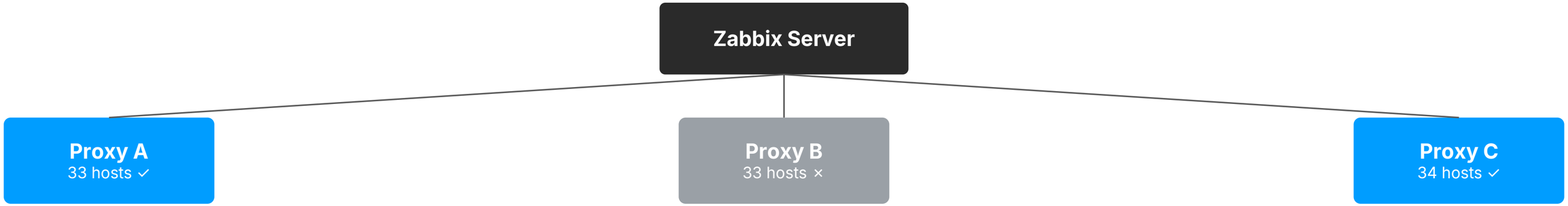
Measured on host count only — not items, not NVPS, not CPU. *(One host with 5,000 items "balances" one with 50.)*

• ⚠ **Hosts on Monitored by = individual proxy are outside this logic entirely — no failover, no rebalance for them.**

State & lifecycle

Three moments that matter in operating a proxy group

Stage 1: Steady state | **Stage 2: Proxy B crashes** | **Stage 3: Proxy B returns**



Zabbix Agent `ServerActive=proxy-a;proxy-b;proxy-c`

1 Steady state: each proxy ~equal load. **2** B crashes: its hosts are pushed to A and C within failover_period. **3** B returns: NO movement until 10× grace elapses and imbalance thresholds trip.

⚠ What the manager *doesn't* see

The only signal driving assignment is **proxy ↔ server** heartbeats. The manager has **no visibility** into **agent ↔ proxy** reachability or **proxy ↔ monitored target** reachability.

Result: a host can be auto-assigned to a proxy that can't reach it (passive) or that its agent can't reach (active) → host appears unavailable, and the manager won't reassign it because, from its point of view, ***everything is fine***. Keep proxies of one group in the same network/zone.

Active agents & Proxy Groups

The slide everyone photographs. Semicolons, not commas.

× COMMA ,

“These are **independent** Zabbix servers / clusters.”

Agent sends each value to **every entry**. Inside one proxy group → ***data duplicated in the DB***.

✓ SEMICOLON ;

“These are **interchangeable entry points** into one cluster.”

Agent picks **any one** that answers, gets back the current proxy list and host→proxy assignments, then talks to its **assigned** proxy — even if that proxy ***isn't in the list***.

```
# zabbix_agent2.conf
ServerActive=proxy-a.example.com;proxy-b.example.com;proxy-c.example.com
```

1
Connect
any listed proxy

2
Receive
current proxy list + load

3
Redirect
to the owning proxy

4
Cache
assignment; fall back on failure

ℹ Only ONE proxy in ServerActive (by accident or design)? ***It still works.***

Even if that proxy isn’t the one assigned to the host: agent connects → receives the full proxy list and current assignments → gets redirected to the right proxy → caches that.

⚠ **Caveat:** if that one listed proxy is offline when the agent starts or restarts, the agent has nobody to talk to → lost data until the proxy comes back. List multiple.

Passive, SNMP, HTTP checks

Cattle, not pets — every proxy must look the same

Passive Zabbix agent
Server= (allowlist, commas OK)

- This is who may connect, not failover semantics
- Proxy that owns the host opens TCP to :10050
- Firewall: every proxy → every agent

SNMP polling
Any group proxy may poll any host

- Firewall: each proxy → each target (UDP :161)
- Often the dominant firewall-opening task in large environments
- Plan the iptables/ACL matrix up front

HTTP / Web / externals
Identical egress from every proxy

- Scripts, ODBC, creds, TLS trust must be identical
- VMware: hosts spread randomly → cache duplication
- Config management is a hard prerequisite

If a check depends on anything local to a proxy — script, driver, credential file, NTP, TLS trust store — it **must be identical.**

zabbix_sender & Proxy Groups

A real gap you need to design around

⚠ **Known gap today:** **zabbix_sender** does **not follow** proxy-group redirects.
If host X is on proxy B but you send to proxy A → A redirects → sender ignores it → data is lost.

OPTION 1

Send to Zabbix Server

Simplest; requires a network path from the sender host.

✓ **Works today**

OPTION 2

Dedicated non-grouped proxy

Point sender/trapper hosts at a proxy that's NOT in a group.

✓ **Clean separation**

OPTION 3

Keep senders off groups

Don't put sender-heavy hosts in proxy groups until fixed upstream.

✓ **Lowest risk**

Check the changelog of newer 7.x minor releases — this is a candidate for a fix. Tracked upstream (same class of problem as python-zabbix-utils #26).

The SNMP traps problem

The one 'but...' the audience will challenge you on

> ***"SNMP traps are not supported by proxies in a proxy group."***
>
> — Zabbix documentation

Why it's hard

Traps are **unsolicited UDP**. The receiving proxy has no way to know which group member currently owns the sending host. There is no redirection model for fire-and-forget UDP.

1

Dedicated non-grouped trap proxy

Simple. But no HA for traps.

2

Accept traps at the server

Central bottleneck. Fine for small estates.

3

Build it yourself

The community pattern on the next slide.

SNMP traps: the community pattern

VIP + duplicator container — known good, don't start from zero

Credit: ICT Open Source Solutions (NL) — blog.zabbix.com • "Proxy group load balancing with SNMP traps"

1

Virtual IP

Keepalived floats a VIP across the proxies. Single, stable trap destination for network devices.

2

Duplicator

Small Docker/Podman container on the VIP holder accepts traps and duplicates them to all other proxies.

3

Owner wins

Every proxy sees every trap. The proxy that currently owns the host processes it; others drop it.

✓ Properties

Single destination IP for devices • fast failover via keepalived • no shared filesystem

× Trade-offs

Intra-group replication cost • traps during VIP flip may be lost (UDP) • extra moving parts

Benefits of Proxy Groups

ZABBIX

Recap — what you actually get

HA at the edge
Finally — without shared storage or VIPs.

Load balancing
Out of the box. Often more valuable than HA itself.

Horizontal scale
Add a proxy → hosts rebalance automatically.

No edge clustering
SQLite / hybrid still perfect. No proxy-DB replication.

Operational simplicity
One group in the UI. One ServerActive per agent.

Safer upgrades
Drain one → group absorbs load → upgrade → reinsert.

Observing the group itself

Monitor the monitor — it (almost) ships out of the box

Zabbix proxy health

per proxy — with agent

RECOMMENDED

Linked to a host that runs a Zabbix agent, monitored by the same proxy it represents.

Why: if the server monitored it, the internal items would report the server's numbers, not the proxy's.

Items: queue, cache, NVPS, processes, last-seen, version.

Remote Zabbix proxy health

per proxy — over TCP/10051

Same content, but uses the proxy's TCP port 10051 directly — no agent on the proxy host required.

Useful from the Zabbix server, or from another Zabbix instance (“monitor the monitoring”).

Zabbix server health

per group — LLD-driven

PER-GROUP

Includes a Proxy group discovery LLD: items + triggers per group (proxies online, total, % availability).

Works out of the box — no manual wiring of zabbix[proxy_group,...] needed.

Monitor

- proxy last-seen
- group online/offline state
- host count per proxy
- NVPS per proxy
- version drift across the group

Dashboard widgets

- host distribution per proxy
- rebalance events timeline
- proxy versions matrix
- per-proxy NVPS outlier panel

Alert on

- group offline
- proxy version drift
- per-proxy NVPS outlier
- rebalance thrash (> N events / day)

⚠ **Required after every major Zabbix upgrade:** ***update the self-monitoring templates*** — Proxy group discovery and the proxy-health items only work if your server's templates match the running version.

Limitations & footguns

The cheat sheet. Photograph this one.

Versions
All proxies 7.0+; same X.Y line as the server (e.g. all 7.0.x). Patches can differ. Older majors → "outdated": data flows, but no config updates.

Single proxy in ServerActive
Works (you get redirected), BUT if it's offline at agent start → no data. List several.

SNMP traps
Not supported natively in Proxy Groups → community pattern (VIP + duplicator).

Network discovery
Discovered hosts go to the proxy GROUP (good). But uniqueness check ignores hosts on individual proxies → duplicates with overlapping subnets.

Small groups (<10)
Distribution will look uneven — expected, documented, not a bug.

Rebalance grace
10 × failover_period before LB fires (~10 min default). Not instant. Not a bug.

ServerActive
Semicolons, not commas. Comma = independent endpoints → duplicated data.

Config parity
Scripts, ODBC, credentials, TLS, NTP — must be identical across all proxies.

zabbix_sender
Use the -g option for multi-host input files; otherwise data may go to the wrong proxy.

VMware monitoring
Hypervisor hosts can land on different proxies → each proxy independently caches from vCenter → multiplied vCenter load.

Asymmetric reachability
Agent can't reach its assigned proxy, but that proxy is fine talking to the server. Manager sees nothing wrong; other proxies redirect agent back to it → loop until the path heals or the proxy itself drops.

Recommendations

Seven opinions. Calibrate to your reality.

01

Fix the stack bottom-up

DB HA → Server HA → Proxy Groups. Skipping layers is theatre.

02

Default to 3 proxies per group

Not a quorum (no shared state) — operational margin: take one down for patching and still have N+1. Below 3 you're either at zero redundancy during maintenance or LB has nothing meaningful to balance.

03

Keep group proxies in one network zone

Manager only watches proxy↔server. If agents and assigned proxies sit in different zones, asymmetric outages create unfixable redirect loops. One group = one site / VLAN / VPC.

04

Treat proxies as cattle

Same config management, same image/AMI, same versions.

05

Standardize agent templates

ServerActive=proxy-a;proxy-b;proxy-c everywhere for that site.

06

Keep a separate ungrouped proxy per site

For SNMP traps, zabbix_sender targets, network discovery.

07

Test failover in non-prod quarterly

HA you never test is HA you don't have.

Q&A / Links

Let's talk. Hallway track continues all day.

Official docs

- **Proxy load balancing & HA**
  `zabbix.com/documentation/current/en/manual/distributed_monitoring/proxies/ha`
- **Server HA**
  `zabbix.com/documentation/current/en/manual/concepts/server/ha`
- **Agent config reference**
  `zabbix.com/documentation/current/en/manual/appendix/config/zabbix_agentd`

Further reading

- Zabbix Blog — Zabbix 7.0 Proxy Load Balancing
- The Zabbix Book — chapter on Proxy Groups
- ICT OSS blog — SNMP traps + proxy groups
  `blog.zabbix.com/proxy-group-load-balancing-with-snmp-traps/31042/`

Christian Anton

Inqbeo — open source consulting & training

Web inqbeo.de
LinkedIn linkedin.com/in/christiananton1/
Matrix @christian.anton:inqbeo.de

Thank you!