A Federated High-Availability Load Balancer for Multi-Site Services in the EGI Federated Cloud

Viet Tran
Institute of Informatics, Slovak Academy of Sciences
viet.tran@savba.sk

IberGrid
Santiago de Compostela, Oct 2026

Single-site high availability stops at the site boundary

Today’s HA tools stay inside one site. Kubernetes clusters, local HAProxy with Keepalived and replicated VMs all fail together with their site.

Full-site outages are not rare. ARGO monitoring of the EGI Federated Cloud shows them regularly.

Instances on several providers solve half the problem. Users still need one address that always leads to a healthy instance.

WHAT IS MISSING

A stable entry point that is not itself tied to any single site.

Goals and non-goals

Goals

- Survive the loss of any single site
- Owners only deploy instances and register
- Active/active or active/standby per service
- Routine changes live within minutes
- Provider-independent and fully open source

Non-goals

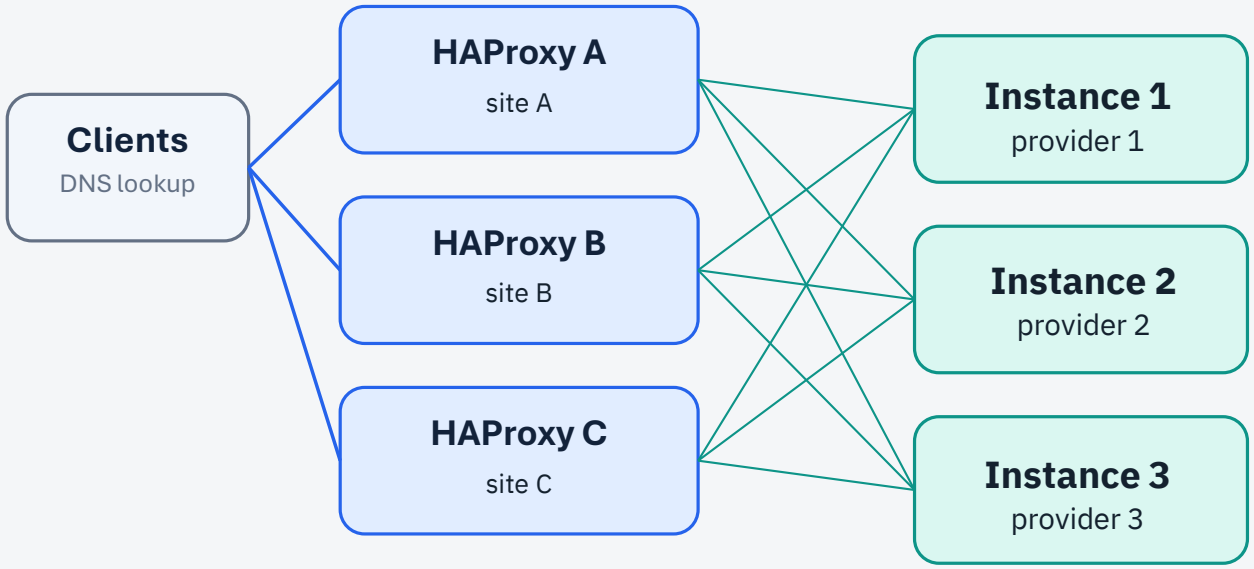
- Replicating data between instances
- Protocols other than HTTP and HTTPS
- Caching or content delivery
- Guarantees when several sites fail at once

Measurable targets replace “no single point of failure”

Target	Proposed value
Frontend availability	99.9% per month
Recovery after losing a node or its site	≤ 3 min for 95% of clients; seconds for most browsers
Detection of a failed backend instance	≤ 20 s
Low-risk change takes effect	≤ 5 min
Review of a new service registration	≤ 2 working days

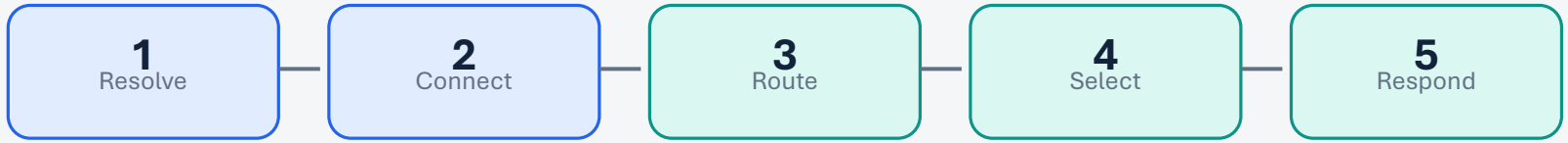
Proposed values, to be agreed in phase 0. All timings in this deck are chosen to meet them.

Conceptual architecture





Control plane • does not carry user traffic



Hostname returns all healthy HAProxy node addresses.

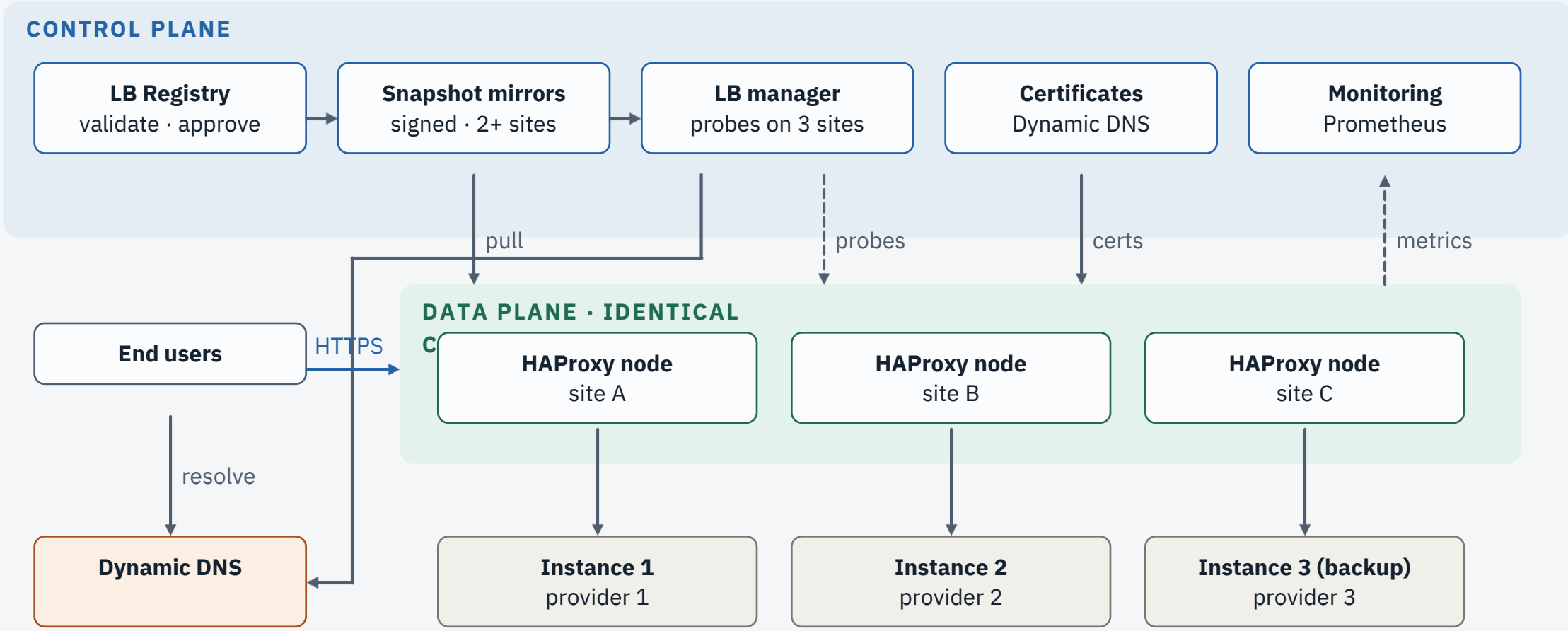
Client connects to one address; retries another if needed.

HAProxy matches hostname and TLS mode.

Healthy backend chosen using the service's balancing policy.

Response returns through the selected HAProxy node.

Detailed architecture



DESIGN RULE

The data plane never needs the control plane to keep serving traffic.

Nodes keep their last valid configuration and certificates. If the registry, the LB manager or monitoring fail, no changes can be made, but every registered service keeps working.

LB Registry: every change follows one lifecycle

1 · Submit
Owner logs in with EGI Check-in and sends YAML

→

2 · Validate
Schema, hostnames, domain proof, haproxy -c

→

3 · Approve
Automatic for routine changes, otherwise an operator

→

4 · Canary
One node applies it and stays healthy for 5 min

→

5 · Stable
All nodes pull, verify, reload and report

→

6 · Active
DNS record created, owner notified

The LB Registry replaces the GitHub repository: same review, history and transparency, plus EGI identities, instant validation and no third-party dependency for the nodes.

Rejected requests return with the operator’s comment. Deleted services are kept for 14 days.

Routine changes skip the queue; risky ones get a review

Approved automatically

- Adding, removing or re-weighting instances
- Marking an instance as backup
- Health-check settings within bounds
- Notification contacts
- Rolling back to an earlier version

Needs an LB operator

- Registering a new service
- Adding a hostname or custom domain
- Changing the TLS mode
- Raising limits above defaults
- Deleting a service

Ownership belongs to an EGI Check-in group or VO entitlement, never a single person, so services are not orphaned when people move on.

Configuration ships as a signed, versioned snapshot

```
{
  "version": 1842,
  "channel": "stable",
  "schema_version": 1,
  "services": [ ... ],
  "signature": "<Ed25519>"
}
```

One signed JSON document holds every active service. Mirrors on 2+ sites serve it if the registry is down.

Each node, every 30 s, on a new version

- Verify the signature
- Render haproxy.cfg and map files
- Validate with haproxy -c
- Reload without dropping connections
- Report the applied version

Any step fails: the node keeps its current configuration. A failed canary stops promotion to stable.

HAProxy data plane: identical nodes on independent sites

Deployment

At least 3 nodes on 3 sites and networks, one image, HAProxy 3.2 LTS, managed as code. Node addresses published for owners' firewalls.

Configuration

Pull every 30 s, render from the snapshot, validate, hitless reload. Backends resolved at runtime.

Health checks

HTTP check every 5 s: down after 3 failures, up after 2. Passive layer-7 checks. Standby instances marked backup.

Shared state and limits

A peers group over TLS shares stick tables, so rate limits hold across nodes. Default connection and rate limits per frontend.

Two TLS modes, chosen per service

	Terminate (default)	**Passthrough**
Where TLS ends	On the HAProxy node	On the backend instance
Load balancer sees traffic	Yes	No
HTTP checks and headers	Yes, plus HTTP/2	No, TCP only
Certificate held by	Central certificate manager	Service owner
Choose it when	Most services	Strict data-protection needs

Both share port 443: a TCP frontend reads the SNI, forwards passthrough services unchanged and hands the rest to an internal terminating frontend.

LB manager: removal by majority, never by one probe

2 of 3

probes, on different sites, must agree for 3 rounds (about 30 s) before a node leaves DNS.

Guard rails

- Healthy = answers and runs a config ≤ 2 versions behind
- Re-added only after 5 healthy rounds
- The last healthy node is never removed
- At most one removal per 2 min without an operator
- Leader elected through etcd; only it writes DNS

DNS publishes all healthy nodes through one pool record

```
pool-default   60  A      <node A>
pool-default   60  A      <node B>
pool-default   60  A      <node C>
service-a     300  CNAME  pool-default
service-b     300  CNAME  pool-default
```

Zone ha.fedcloud.eu, owned by the LB operators; the LB manager is its only writer.

- Failover changes one record, not one per service
- Clients get all healthy nodes and retry the next
- Custom domains: CNAME to the service name (no zone apex)
- Hostnames: letters, digits, hyphens; no underscores

> Authoritative DNS must run on 2+ independent networks: failover depends on it completely.

Losing a node: what users experience

0 s
Node A or its site fails

→

~30 s
2 of 3 probes confirm, 3 rounds

→

~35 s
Leader removes A from the pool record

→

≤ ~95 s
Cached records expire (TTL 60 s)

Seconds
Clients that retry the next address on a failed connection, including most browsers

1–2 minutes
Clients that do not retry; some resolvers and Java runtimes cache longer

Certificates and alerts are handled centrally

Certificates

- No private keys in registrations
- Central ACME client with DNS-01
- Wildcard *.ha.fedcloud.eu; custom domains by delegated validation
- Keys in a secrets store, read-only per node
- Renew at 30 days left, alert at 14

Monitoring and alerts

- Built-in Prometheus metrics on every node
- 2 Prometheus replicas plus Alertmanager
- Backend alerts only when ≥ 2 nodes agree
- One notification per event, not per node
- Owner dashboards and ARGO integration

Every failure has a defined response

Failure	Response	Effect on users
Backend instance fails	Out of rotation in ~15 s; owner alerted	In-flight requests may fail
HAProxy node or site fails	Removed from the pool in ~30 s	Seconds to 1–2 minutes
Network split, one probe	No majority, no action	None
Registry or LB manager fails	Nodes keep their snapshot; new leader	None; changes delayed
Faulty configuration	Canary stops promotion; rollback	At most the canary node
Dynamic DNS API down	Retries; operators alerted	No failover; clients still retry

What service owners must provide

- **Instances on 2+ providers**, 3 for active/active
- **A real health endpoint** that checks dependencies
- **Works under the frontend name**: links, cookies, OIDC redirect URIs
- **Uses forwarded headers** or the PROXY protocol
- **Accepts traffic only** from published node addresses
- **Replicates or shares its data**, or enables stickiness
- **Valid backend TLS certificate**
- **Submits one short registration** in the LB Registry

Why HAProxy rather than Traefik

Criterion	HAProxy	Traefik (open source)
Config from a registry	Generated, checked by haproxy -c	Native YAML, no strict check
State shared across nodes	peers: shared rate limits	None, limits per instance
Health checks	Very mature, active and passive	Active and passive in 3.x
Certificates on many nodes	Central ACME either way	Built-in ACME is per instance
Resource use per VM	Lower	Higher at high load

Chosen: HAProxy. Registrations stay proxy-neutral, so a switch needs only a new template. Traefik would win if nodes moved to Kubernetes. Anycast and commercial GSLB were rejected for cost, lock-in and the need for a shared ASN.

Advantages and limitations

Advantages

- Survives the loss of any single site
- HA and load balancing in one service
- Owners deploy and register, nothing more
- Monitoring and alerts included
- No provider-specific features; open source

Limitations

- Slow recovery for clients that ignore TTLs
- Depends on highly available DNS
- HTTP and HTTPS only
- Extra network hop: latency and bandwidth
- Terminate mode exposes decrypted traffic

Thank you for your attention

Discussion

Backup slide
for service owner: keeping data in sync across sites

Strategy	Consistency across sites	Examples	LB mode
Consensus (Raft), distributed SQL	Strong; writes wait for a majority	YugabyteDB, TiDB, CockroachDB	Active/active
Primary–replica, automated failover	Strong in one region; async DR	PostgreSQL + Patroni, MongoDB	Active/standby
Leaderless, tunable	Eventual; quorum per site	Cassandra, ScyllaDB	Active/active
Multi-master with CRDTs	Converges by merging	Redis Enterprise, CouchDB	Active/active
Replicated event log	Each site rebuilds its state	Kafka + MirrorMaker 2, NATS	Either
Object and file replication	Asynchronous	Ceph RGW multisite	Either

For owners: use 3+ sites (or a witness) for quorum, split data by the consistency it needs, and rehearse losing a whole site. The load balancer routes traffic; it never replicates data.