Verae Time × Zapier — system overview

High-level description of the whole system: what it is, which **independent git repositories** implement it, how it stays up, how it behaves when the local network fails, and how it talks to the **central Verae NATS.IO 3-server cluster**. New work is added by **new address routing**, not by teaching Zapier about NATS.

This repo: https://git.georgelambert.org/marchon/overview
Live catalog (PDF by default): https://zapier.georgelambert.org/ · Markdown indexes
Clone: `ssh://git@git.georgelambert.org:2223/marchon/overview.git`

Table of contents

One-screen picture

System context

System context

Zapier → HTTPS `zappier-edge` → HTTPS `verae-middleware` → **NATS cluster** → workers, WORM archives, tree nodes. Chain HTTPS is `api.veraetime.net` (or MOCK). Fleet keeps minimum copies, including tree nodes.

Independent repositories (Forgejo)

Repo	Role
overview	This document
verae-ops	Docker, Proxmox, VMs, dedicated hardware
verae-nats-process	**Model** for a new addressed process
master-zapier-plan-draft	Monorepo snapshot
zappier-edge	Metered public HTTPS
verae-middleware	Zapier HTTP + NATS workers
verae-zapier-app	Zapier Platform app
verae-activate	Activate-now app
verae-request-splitter	Hash vs attachments
verae-archive-worm	Bloom WORM node
verae-archive-aggregator	Archive reply merge
verae-tree-node	Merkle leaf proofs
verae-fleet	Replica floor + SSH hosts
verae-zapier-simulator	Trace console
zapier-user-docs	Signup → lookup
zapier-docs-master	Per-module SUMMARY + NATS

Clone any of them: `git clone ssh://git@git.georgelambert.org:2223/marchon/<name>.git` (SSH port **2223**).

Operator console (loopback)

Fleet, message **Trace**, and **Docs** share one shell at http://127.0.0.1:3850/ (`verae-fleet`). Not public. See CONSOLE.pdf.

- Fleet tab
- Fleet tab

Trace tab
Trace tab

Docs tab
Docs tab

Where to read next

Start at the top. Architecture is second. Everything else is reference.

1. High-level (non-technical)

Document	What it is
This overview	Whole system in one place
Modules and repositories	Short names of every independent repo
zapier-docs-master	One-line SUMMARY + NATS per module
zapier-user-docs	Signup → register a hash → lookup

2. Architecture

Document	What it is
NATS 3-server cluster	Who may connect; private URLs
Uptime	Replica floors, restart, SSH hosts
Local network failures	Reconnect, bloom silence, failover
Address routing	Unplanned functions as new subjects
External resources	Chain, WORM, tree-node search
Diagrams	SVG + mermaid
Fleet	Operator replica control
Module NATS map	Address table
Archive / bloom	Multipart and multi-receipt
Tree nodes	Bulk Merkle leaves
Composition	zappier + middleware HTTPS

3. Remaining documentation

Document	What it is
Documentation index	A–Z and catalog URLs
Expansion template / verae-nats-process	Copy this to add an address
Operator console	Screenshots of Fleet / Trace / Docs
Simulator	Standalone trace (also inside the console)
Zapier developer setup	CLI / login / push
Live catalog	https://zapier.georgelambert.org/ (PDF) · Markdown indexes

Credits

Design: **Scott Lindsey**, **George Lambert**, **NATS.IO**, and **Grok-Code** by Grok.com.

verae-ops — install, run, maintain

How to stand up Verae Time × Zapier on **Docker**, **Proxmox**, **generic VMs**, or **dedicated hardware**, and how to **link dependent services**.

Forgejo: https://git.georgelambert.org/marchon/verae-ops
Clone: `ssh://git@git.georgelambert.org:2223/marchon/verae-ops.git`
Catalog: https://zapier.georgelambert.org/packages/verae-ops/README.pdf
System map: https://zapier.georgelambert.org/overview/README.pdf

This is the operations repo. Application code lives in the other independent git repositories.

Reading order

1. 01-dependencies.md — what must exist, who talks to whom
2. 02-docker.md — Compose (NATS 3-node cluster + HTTPS edges)
3. 03-proxmox.md — LXC / QEMU VMs
4. 04-virtual-servers.md — cloud or hypervisor VMs
5. 05-dedicated-hardware.md — bare metal (NS1-style)
6. 06-linking-services.md — env vars, URLs, replica floors
7. 07-maintenance.md — upgrade, backup, fleet, NATS

Public PDFs of the same files: `https://zapier.georgelambert.org/packages/verae-ops/<name>.pdf`.

Non-negotiables

Rule	Why
Zapier cloud → **HTTPS only** (zappier-edge)	Never NATS, never `api.veraetime.net`
NATS binds **loopback or a private docker/VM net**	Not on the public NIC
Tree-node **min 3**, pause does not count	Bulk Merkle lookups
Private keys stay on disk; git stores **paths**	`machines.json identityFile`
Operator console is **loopback** `:3850`	Not a public site

Quick lab (one machine)

```
git clone ssh://git@git.georgelambert.org:2223/marchon/verae-ops.git
cd verae-ops
docker compose up --build
```

Then: zappier http://127.0.0.1:3000/ middleware http://127.0.0.1:3100/health NATS monitoring http://127.0.0.1:8222/

Production layout is three NATS nodes + fleet-spread workers; see 02-docker.md and 05-dedicated-hardware.md.

Zappier

Metered API platform: per-endpoint pricing, customer types with multipliers and monthly credits, a usage ledger, Stripe metered billing, purchase-order invoicing, a company admin console, a self-service customer portal, and a Zapier integration.

Forgejo: https://git.georgelambert.org/marchon/zappier-edge
Catalog README: https://zapier.georgelambert.org/packages/zappier/README.pdf

Documentation (public PDFs)

- USER-MANUAL — operations & usage
- ACCOUNTING — invoices, PO billing, reports

- USER-MANAGEMENT — pricing, customer types
- CUSTOMER-PORTAL — signup, 2FA, reloads
- DEVELOPER
- WALKTHROUGH
- Markdown copies remain next to these files in `docs/`
- **Sample Zapier app:** `zapier-app/` — https://zapier.georgelambert.org/packages/zappier/zapier-app/README.pdf (if present) or the source tree in git

Surfaces

Surface	URL	Audience
Public API	`/v1/*`	API customers (`x-api-key`)
Interactive API docs	`/docs`	Integrating developers
Admin console	`/admin`	Company ops & accounting
Customer portal	`/portal`	End-user customers (signup, 2FA, billing)
Zapier app	`zapier-app/`	No-code users via Zapier

Pricing model

`openapi.yaml` defines the API surface; each `operationId` is a rate-card key. Endpoints carry **list prices** (seed: `src/pricing.ts` → `DEFAULT_RATE_CARD`). Customer types are **tier configs** (`DEFAULT_TIERS`) with a `multiplier` , a `monthlyCreditCents` quota, and an optional `defaultRule` for endpoints not on the card. Individual customers can carry a `multiplierOverride` . Billed price = `round(list price × multiplier)` ; usage up to the monthly credit is free. Pricing is editable at runtime in the admin console.

Seed rate card (list prices, cents per call)

Endpoint	Model	List price
`status`	free	0
`storage-list`	free	0
`transform`	fixed	4
`storage`	variable	10 + 1 per KB metadata + 50 per MB attached

Seed customer types

Tier	Multiplier	Monthly credit	Default rule (unlisted endpoints)
`free`	1.0	100 cents	none — call rejected with 403
`pro`	0.5	1000 cents	fixed 8 list → 4 billed
`business`	0.25	10000 cents	fixed 8 list → 2 billed

Adding a new API call = add it to `openapi.yaml`, then price it in the admin UI. Adding a customer type = create it in the admin UI. Variable pricing = base per call + metadata size (rounded up to KB) + attachment size (rounded up to MB), then the multiplier.

Quickstart

```
npm install
npm run dev
```

The server starts on port 3000. API docs at http://localhost:3000/docs, admin console at `/admin`, customer portal at `/portal`.

Deploying

```
npm ci && npm run build
node dist/index.js        # runs from ANY working directory
```

All runtime paths (SQLite default, `.env`, OpenAPI spec, static assets) resolve from the installation root, so the compiled server works under systemd, Docker, or cron regardless of cwd. `PORT` and `ZAPPIER_DB` remain environment-overridable.

Environment variables

Variable	Default	Purpose
`PORT`	`3000`	HTTP port the server listens on
`ZAPPIER_DB`	`<root>/zappier.db`	SQLite database file path
`ADMIN_KEY`	`admin-dev-key`	Admin UI / admin API key — **set a real secret in production**
`ADMIN_USER`	`admin`	Admin UI primary login username
`DEMO_ADMIN_USER`	`demo`	Admin UI demo login username
`DEMO_ADMIN_PASSWORD`	`$$$Adm1n###`	Demo login password — **override in production**
`STRIPE_SECRET_KEY`	*(none)*	Stripe secret key — billing job and portal reloads

Billing

Usage is reported to Stripe by a job (loads `STRIPE_SECRET_KEY` from `.env`):

```
npx ts-node src/jobs/report-usage.ts
```

The job sums each customer's usage since the first of the current month (UTC), applies the tier's monthly credit, and reports only the **delta** above what was already reported — re-runs are safe. Idempotency comes from three layers: a `billing_reports` ledger (cumulative cents per customer per month), an atomic `job_locks` run guard (1 h TTL), and a deterministic Stripe event `identifier` (`customer:period:billable`) that dedupes crash retries. It requires a Stripe meter named `zappier.api_cents` with Sum aggregation over the `value` field.

A Kimi cron job ("Zappier billing · report usage to Stripe") runs it daily at 06:17 America/New_York with a completion notification.

Purchase-order customers are invoiced manually from the admin console (**Invoices** tab); prepaid balances from the customer portal are drawn down automatically at invoice issue. See `docs/ACCOUNTING.md`.

Zapier app

The companion Zapier integration lives in `zapier-app/`:

```
cd zapier-app
npm install
npm test
```

To deploy it, create a Zapier developer account, run `zapier login`, then `zapier push` from the `zapier-app/` directory.

Testing

```
npm test                    # root API/service suite (jest, 165 tests)
cd zapier-app && npm test   # Zapier integration suite (mocha, 4 tests)
```

verae-middleware

Zapier-facing HTTPS adapter (`/zapier/v1/*`) plus NATS workers. Zapier never speaks NATS or `api.veraetime.net`; this process does.

Forgejo: https://git.georgelambert.org/marchon/verae-middleware
Clone: `ssh://git@git.georgelambert.org:2223/marchon/verae-middleware.git`
Catalog README (PDF): https://zapier.georgelambert.org/packages/verae-zapier-middleware/README.pdf
NATS contract: https://zapier.georgelambert.org/docs-master/modules/verae-zapier-middleware/NATS.pdf
Install / Docker / Proxmox: https://zapier.georgelambert.org/packages/verae-ops/README.pdf

Run

```
cd packages/verae-zapier-middleware
npm install
MOCK_VERAE=true NATS_ENABLED=false npm test
PORT=3100 MOCK_VERAE=true npm start   # /health, /zapier/v1
```

Docker image: `Dockerfile` (Node 22, port 3100). Full stack: `verae-ops` compose.

Environment

Variable	Default	Purpose
`PORT`	`3100`	HTTP bind
`VERAE_API_BASE_URL`	`http://localhost:8080`	Chain HTTPS
`MOCK_VERAE`	`false`	In-process mock chain
`NATS_ENABLED`	`false`	JetStream vs in-process poller
`NATS_URL`	`nats://127.0.0.1:4222`	Cluster URL (loopback / docker DNS)
`WAIT_TIMEOUT_MS`	—	Sync wait on `jobs.events`

Depends on

- **Upstream HTTPS:** zappier-edge (`x-api-key` already checked) or a lab client
- **Chain:** `api.veraetime.net` or `MOCK_VERAE=true`
- **NATS:** `jobs.watch` out, `jobs.events` in; optional `archive.put` / aggregator wait
- **Optional:** request-splitter (in-process), archive-worm, tree-node, webhook-deliver

Does **not** bind NATS publicly. See verae-ops linking.

verae-zapier

Zapier Platform CLI app for Verae. Runs on **Zapier's servers**. Calls only middleware HTTPS (`MIDDLEWARE_BASE_URL`), never NATS and never `api.veraetime.net` .

Forgejo: https://git.georgelambert.org/marchon/verae-zapier-app
Catalog: https://zapier.georgelambert.org/packages/verae-zapier/README.pdf
Sphinx API sheets: https://zapier.georgelambert.org/sphinx/index.html

Role

Runs on **Zapier's servers**. Calls only the middleware HTTPS API (`MIDDLEWARE_BASE_URL`), never NATS and never `api.veraetime.net` directly.

Planned modules

File	Purpose
`authentication.js`	Custom API key auth → `GET /zapier/v1/auth/me`
`index.js`	App definition, beforeRequest, afterResponse error mapping
`creates/timestamp_and_wait.js`	Primary action
`creates/create_timestamp.js`	Async jobId action
`creates/verify_timestamp.js`	Verify certificate
`creates/batch_timestamp.js`	Batch create
`searches/job_status.js`	Lookup by jobId
`triggers/timestamp_completed.js`	REST Hook

I/O contracts: https://zapier.georgelambert.org/docs/developer/modules/function-reference.pdf

Env

```
export MIDDLEWARE_BASE_URL=https://your-middleware.example.com
```

Gate

```
npm run gate:11   # from monorepo root, after Phase 11 implementation
```

Verae Time — Zapier activate app

This is the integration you **push tomorrow**. It has one action:

Add Numbers — `number1` + `number2` → `sum`.

It runs **inside Zapier**. You do not need `api.veraetime.net`, NATS, or a public URL.

Catalog: https://zapier.georgelambert.org/packages/verae-activate/README.pdf
Setup (PDF): https://zapier.georgelambert.org/docs/04-activate/SETUP-ZAPIER-DEVELOPER.pdf
Sample action: `creates/add_numbers.js` — public sheet https://zapier.georgelambert.org/docs/modules/verae-activate/creates/add_numbers.pdf

Step-by-step account setup: SETUP-ZAPIER-DEVELOPER

```
cd packages/verae-activate
npm install
npm test
zapier-platform login
zapier-platform register "Verae Time"
zapier-platform push
```

verae-request-splitter

Splits a Zapier/middleware body into (1) the **chain hash** and (2) off-chain **archive puts**. File bytes never go on `api.veraetime.net`.

Forgejo: https://git.georgelambert.org/marchon/verae-request-splitter
Catalog: https://zapier.georgelambert.org/packages/verae-request-splitter/README.pdf
NATS: https://zapier.georgelambert.org/docs-master/modules/verae-request-splitter/NATS.pdf

Run

```
cd packages/verae-request-splitter
npm test
```

Today middleware calls this **in-process**. A later split-out can subscribe `verae.splitter.in`.

Outputs

Path	Content
Chain	`{ sha256, hashAlg }` only
`verae.archive.put`	each `publicMeta` / `privateMeta` / `file`
Batch	Merkle **root** on chain; leaves → tree-node puts (`kind=tree`)

Depends on

- middleware-http (caller)
- archive-worm / tree-node (puts)
- chain client (hash seal)

Does not talk to Zapier or the job poller.

verae-archive-worm

Bloom-filtered append-only WORM node. Stores metadata and files **off chain**. A bloom **miss is silence** (no NATS reply).

Forgejo: https://git.georgelambert.org/marchon/verae-archive-worm
Catalog: https://zapier.georgelambert.org/packages/verae-archive-worm/README.pdf
NATS: https://zapier.georgelambert.org/docs-master/modules/verae-archive-worm/NATS.pdf

Run

```
cd packages/verae-archive-worm
npm test
```

Fleet keeps **min 3** copies (`keepFloor`). Pause does not count as available.

Addresses

Subject	Role
`verae.archive.put`	Append `{ sha256, kind, record }` and add the hash to the bloom
`verae.archive.query`	Broadcast (no queue group). Hit → `verae.archive.reply.<correlationId>`
miss	**no message**

Kinds: `publicMeta` , `privateMeta` , `file` . Tree leaves live on **verae-tree-node** (`kind=tree`).

Depends on

- NATS JetStream reachable on the **private** URL (`NATS_URL`)
- archive-aggregator (query fan-out / merge)
- fleet replica floor

Does not talk to Zapier or the chain.

verae-archive-aggregator

When a wait path sets `includeAttached` , this process broadcasts `verae.archive.query` and **merges** replies into the job JSON. Silent bloom misses omit slices; the chain receipt still succeeds.

Forgejo: https://git.georgelambert.org/marchon/verae-archive-aggregator
Catalog: https://zapier.georgelambert.org/packages/verae-archive-aggregator/README.pdf
NATS: https://zapier.georgelambert.org/docs-master/modules/verae-archive-aggregator/NATS.pdf

Run

```
cd packages/verae-archive-aggregator
npm test
```

Addresses

Direction	Subject
out	`verae.archive.query` `{ correlationId, sha256, tenantId, kinds[] }`
in	`verae.archive.reply.<correlationId>` (hits only)

Depends on

- middleware wait path (`includeAttached` / `includeTree`)
- one or more archive-worm and/or tree-node replicas
- NATS (private)

Timeout: missing archives omit their records; do not fail the seal.

verae-tree-node

Bloom-filtered WORM store for Merkle **leaf** proofs. Only the Merkle **root** is sealed on the chain.

Forgejo: https://git.georgelambert.org/marchon/verae-tree-node
Catalog: https://zapier.georgelambert.org/packages/verae-tree-node/README.pdf
User lookup guide: https://zapier.georgelambert.org/user-docs/09-lookup-tree-nodes.pdf

When Zapier submits a **batch**, middleware builds a Merkle tree of item SHA-256s, seals **only the root** on `api.veraetime.net`, and puts each leaf proof on a sharded tree node. A later lookup of a leaf that is not on the main chain fans out `verae.archive.query` with `kinds: ["tree"]`. Nodes that do not hold the leaf send **nothing**.

Run

```
cd packages/verae-tree-node
npm test
```

Fleet **keepFloor min=3**. Pause does not count toward the floor.

Addresses

Same as archive-worm, with `kind=tree`: `verae.archive.put` / `query` / `reply.<id>` (hit only).

Depends on

- NATS (private)
- archive-aggregator for bulk lookup
- fleet replica floor
- chain holds the root only

Clone: `ssh://git@git.georgelambert.org:2223/marchon/verae-tree-node.git`

verae-fleet

Operator control plane for Verae Time × Zapier **runtime** services:

- a **list** of every service and its config file
- a **central replica spec** (`fleet.json`) — how many copies must be up
- a **monitor** of active / paused / unhealthy replicas
- **restart** when a copy is offline or fails `/health`
- **on / off / pause / resume** per service or per instance
- **keepFloor** on tree nodes so at least `min` copies stay available (paused copies do not count)

```
cd packages/verae-fleet
npm test
node src/cli.js list
node src/cli.js serve    # http://127.0.0.1:3850/
```

Against a running daemon:

```
node src/cli.js status
node src/cli.js pause tree-node-0      # floor starts another tree-node
node src/cli.js resume tree-node-0
node src/cli.js restart tree-node-1
node src/cli.js stop webhook-deliver   # disable that service
node src/cli.js start webhook-deliver
```

Add capacity in `machines.json` or the monitor **Add machine** form (`kind=ssh`, user, host, identity file path). New replicas land on the least-loaded eligible host.

```
node src/cli.js ssh-check ns1    # marchon@70.88.205.138 with ~/.ssh/id_ed25519
```

Private keys stay on disk (`~/.ssh/id_ed25519`); git stores only the path. Optional overrides: `machines.secrets.json` (gitignored).

Operator console (Fleet · Trace · Docs) at http://127.0.0.1:3850/ — docs/CONSOLE.md · CONSOLE.pdf (also https://zapier.georgelambert.org/packages/verae-fleet/docs/CONSOLE.pdf). Message-processing **min / avg / p50 / p90** RTT is on the Fleet tab.

Zapier cloud apps are listed but **not spawned**. NATS on NS1 is **monitored only** (loopback `:4222`, never a public bind).

Install on Docker / Proxmox / metal: https://zapier.georgelambert.org/packages/verae-ops/README.pdf

Clone: `ssh://git@git.georgelambert.org:2223/marchon/verae-fleet.git`

Verae Zapier interface simulator

In-process stand-in for the Zapier editor. Run a step, then read a **trace console** of every hop from Zapier input through zappier-edge, middleware, splitter, mock chain, WORM archives, and tree nodes, back to the Zapier output.

Use it to validate messaging and to catch errors, delays, failures, and recoveries **before** the Zapier app is pushed live.

```
cd packages/verae-zapier-simulator
npm test
npm start   # http://127.0.0.1:3847/
```

What it simulates

Zapier action	Path
Sign up	zappier portal → API key
Connect	Zapier test auth (`x-api-key`)
Create Timestamp (async)	202 `jobId` + `jobs.watch`
Create Timestamp and Wait	hold until `jobs.events`
Find Timestamp by SHA256	central Verae chain only
Find Hash (tree nodes)	chain miss → broadcast `verae.archive.query` kinds= `tree`
Create Batch	Merkle root sealed on chain; leaves on tree nodes
Verify	mock certificate

Faults you can inject: edge 401/402, chain timeout, NATS watch/events drop, archive put fail, all archives silent, one tree node down, hop delays, then **recover** (retry).

The monitor panel flags broken assumptions (Zapier talking to NATS, file bytes on chain, bloom miss sending a reply) and suggests changes (tree-node search, wait+hook pairing, 402 upgrade URL, archive quorum).

Catalog: https://zapier.georgelambert.org/packages/verae-zapier-simulator/README.pdf
Live sample UI (loopback): http://127.0.0.1:3847/ — also inside the operator console Trace tab.
Static copy: https://zapier.georgelambert.org/simulator/

Clone: `ssh://git@git.georgelambert.org:2223/marchon/verae-zapier-simulator.git`

Verae Time for Zapier — user guide

How to go from **zero account** to **registering SHA-256 hashes**, **looking them up on the central Verae chain**, and **finding hashes that were only sealed as part of a bulk Merkle summary** on external tree-node archives.

This is the customer-facing guide (signup → Zaps). Developers: see zapier-docs-master and the simulator at verae-zapier-simulator.

Live catalog: https://zapier.georgelambert.org/
Clone: `ssh://git@git.georgelambert.org:2223/marchon/zapier-user-docs.git`

Contents

Try it without going live

The interface simulator is a Zapier-like form plus a trace console. Run **Batch Merkle**, copy a leaf SHA-256, then **Find Timestamp by SHA256** (miss) vs **Find Hash (tree nodes)** (proof).

zapier-docs-master

Summaries, NATS contracts, and message flows for every Verae Time × Zapier module.

Live catalog: https://zapier.georgelambert.org/
Overview (start here): https://git.georgelambert.org/marchon/overview
Operator console: http://127.0.0.1:3850/ (Fleet · Trace · Docs)
Monorepo: https://git.georgelambert.org/marchon/master-zapier-plan-draft (`main` and `master`)

Git repos (Forgejo on NS1)

Repo	URL
master-zapier-plan-draft	https://git.georgelambert.org/marchon/master-zapier-plan-draft
zappier-edge	https://git.georgelambert.org/marchon/zappier-edge
verae-middleware	https://git.georgelambert.org/marchon/verae-middleware
verae-zapier-app	https://git.georgelambert.org/marchon/verae-zapier-app
verae-activate	https://git.georgelambert.org/marchon/verae-activate
verae-request-splitter	https://git.georgelambert.org/marchon/verae-request-splitter
verae-archive-worm	https://git.georgelambert.org/marchon/verae-archive-worm
verae-archive-aggregator	https://git.georgelambert.org/marchon/verae-archive-aggregator
verae-tree-node	https://git.georgelambert.org/marchon/verae-tree-node
verae-zapier-simulator	https://git.georgelambert.org/marchon/verae-zapier-simulator
zapier-user-docs	https://git.georgelambert.org/marchon/zapier-user-docs
verae-fleet	https://git.georgelambert.org/marchon/verae-fleet
overview	https://git.georgelambert.org/marchon/overview
verae-nats-process	https://git.georgelambert.org/marchon/verae-nats-process
zapier-docs-master (this repo)	https://git.georgelambert.org/marchon/zapier-docs-master

Repo	URL
verae-ops	https://git.georgelambert.org/marchon/verae-ops

Clone (SSH port 2223):

```
git clone ssh://git@git.georgelambert.org:2223/marchon/<name>.git
```

Modules

Module	Summary	NATS in	NATS out
zappier-edge	Metered HTTPS for Zapier	—	— (HTTPS to middleware)
verae-middleware	Job id + wait HTTP	jobs.events	jobs.watch
verae-activate	Pushable Zapier app	—	—
verae-zapier-app	Full Zapier nouns	—	—
request-splitter	Hash vs files	—	archive.put
archive-worm	Bloom WORM node	archive.query, archive.put	archive.reply.* (hit only)
archive-aggregator	Merge archive replies	archive.reply.*	archive.query
tree-node	Merkle leaf proofs	archive.query, archive.put	archive.reply.* (hit only)
zapier-simulator	Trace console (in-process)	—	—
zapier-user-docs	Signup → lookup guide	—	—
verae-fleet	Replica floors + monitor	—	—
overview	System map	—	—
nats-process	Template worker	example.process.in	example.process.out / reply.*
ops	Install / Docker / Proxmox / metal	—	—

Documents in this repo

- MESSAGE-FLOWS.md — numbered request paths
- modules-and-nats.md — address table
- archive-nats.md — bloom / multi-receipt design
- composition.md — HTTPS hops

- `modules/<name>/SUMMARY.md` and `NATS.md`

verae-nats-process

Template for a new **addressed messaging process** on the central Verae NATS.IO cluster.

Repo: https://git.georgelambert.org/marchon/verae-nats-process
Clone: `ssh://git@git.georgelambert.org:2223/marchon/verae-nats-process.git`
Overview: https://zapier.georgelambert.org/overview/README.pdf
Routing: https://zapier.georgelambert.org/packages/verae-nats-process/ROUTING.pdf

```
npm test
npm start   # /health on 127.0.0.1:13900
```

1. Duplicate this repository (new Forgejo name).
2. Edit `src/subjects.js` — `verae.<area>.<resource>.*`.
3. Replace `handle()` in `src/handle.js`.
4. Update `ROUTING.md`; add a fleet service with `min` / `max`.
5. If Zapier needs the result, add HTTPS on **verae-middleware** only.

Zapier cloud must not subscribe. See address routing.