

dev-overview — NATS routes and developer credentials

dev-overview

Developer map of the Verae Time × Zapier **NATS message fabric**: what was designed, what is actually running on NS1, how a Zapier timestamp job reaches `api.veraetime.net` (or the mock) and comes back, and what is still unwired.

Forgejo: <https://git.georgelambert.org/marchon/dev-overview>

SSH: `ssh://git@git.georgelambert.org:2223/marchon/dev-overview.git`

Catalog: <https://zapier.georgelambert.org/packages/dev-overview/>

Date: 2026-09-14

Zapier cloud **never** connects to NATS. It talks HTTPS to `https://zap.zapier.georgelambert.org` (access-zapier) then middleware. `api.veraetime.net` is **HTTPS only** — middleware is the NATS client.

Contents

Doc	What
00-sources.md	Every other doc that names a NATS address or TCP route
01-nats-routes.md	Transport ports + every <code>vera.*</code> address: designed vs running
02-endpoint-io.md	Per-address: purpose, inbound from, outbound to, how to connect Verae API
03-zapier-verae-path.md	Zapier → middleware → Verae timestamp server → poll → events → Zapier
04-gaps-and-finish.md	Modules not connected; billing/event/aux; ordered finish list
05-developer-credentials.md	

Doc	What
	URLs, tokens, lab API keys, staff/IAM, git, env cheat sheet

Cluster (running)

nats://10.10.10.21:4222,nats://10.10.10.22:4222,nats://10.10.10.23:4222

JetStream streams (replicas=3, file): ZAPIER_JOBS, ZAPIER_EVENTS, ZAPIER_WEBHOOKS, ZAPIER_USAGE, VERAЕ_ARCHIVE.

Public token-gated **client** edge (not cluster routes): nats://<token>@70.88.205.138:4222,wss://nats.zapier.georgelambert.org. Do not put this in Zapier.

Sources (other documentation that names routes)

This inventory is compiled from code and from these docs. Do not invent extra verae.* names.

Master maps

Path	What it names
packages/docs-master/modules-and-nats.md	Module catalog, subject table, HTTPS vs NATS
packages/docs-master/MESSAGE-FLows.md	Create/wait/includeAttached/batch Merkle
packages/docs-master/archive-nats.md	Archive put/query/reply
docs/02-architecture/nats-subjects.md	verae.zapier.* payloads + streams
docs/02-architecture/	Same catalog as docs-master

Path	What it names
modules-and-nats.md	
docs/02-architecture/overview.md	Poller / events / webhooks
packages/overview/03-nats-cluster.md	Cluster URLs, who may connect
packages/overview/06-address-routing.md	Address pattern + plane mapping
packages/verae-ops/06-linking-services.md	How services attach
packages/verae-nats-cluster/scripts/ensure-streams.sh	Live stream create
packages/verae-nats-cluster/conf/nats.conf.tpl	:4222 / :6222 / :7422 / monitor :8222
packages/verae-nats-cluster/conf/public-edge.conf.tpl	Public leaf :4222, WS :14422
packages/verae-fleet/overlays/ns1/fleet.json	Enabled vs in-process workers
packages/verae-keep units.px-worker.json	Worm/tree on LXC 510
packages/verae-zapier-middleware/src/nats/subjects.js	Product subjects + consumers
packages/zappier-account-balance/src/subjects.js	Billing subjects

Path	What it names
packages/verae-access-authz/src/subjects.js + policy.js	Authz + plane allow lists
packages/zapper/src/billing-nats.ts	Edge billing client

Per-package NATS.md (Forgejo module copies also live under docs-master/modules/)

Package	Addresses
verae-zapper-middleware	jobs.watch out, jobs.events in, archive.query via aggregator
verae-jobs-events	jobs.events; exclusive → verae.internal.jobs.events
verae-request-splitter	archive.put
verae-archive-worm	archive.put / query / reply.<id>
verae-tree-node	same, kind: tree
verae-archive-aggregator	query out, reply.<id> in
zapper-account-balance	verae.billing.*
zapper (edge)	package NATS.md is stale (“does not publish”); code publishes usage/payment/credit/customer.put when NATS_URL is set. Use docs-master/modules/zapper/NATS.md
zapper-customer-service	statement.get, balance.adjust
zapper-sales-pricing	statement.get
zapper-accounting-export	statement.get
zapper-identity	identity.bind / lookup
verae-access-authz	authz.check / authz.deny
verae-access-web	access.web.in / access.web.billing.*
verae-access-api	access.api.*
verae-access-zapper	access.zapper.* then HTTPS to middleware
verae-access-leaf	access.leaf.in

Package	Addresses
verae-access-staff	access.staff.*
verae-nats-process	verae.example.process.* (template)
verae-nats-accounts	policy only (INTERNAL vs LEAF)
verae-zapier-simulator	in-process replay of product addresses
verae-keep / verae-fleet / verae-uptime / verae-backup	no product subjects

Simulator `src/pipeline.js` and `src/monitors.js` list the same product addresses for traces; they are not live subscribers.

Designed vs running NATS routes

Designed = named in docs or in code.

Running = a process on NS1 is subscribed or publishing it **today**, and (for durable work) a JetStream stream exists.

Cluster name `verae`. Internal clients use the three LXC on `vmbr1`. Public edge is a **leaf**, not a second cluster.

Legend: **JS** = JetStream stream; **RR** = core request-reply; **pub** = core or JS fire-and-forget; **Q** = queue group.

Transport (TCP / leaf / public)

These are NATS **server** routes, not `verae.*` subjects.

Endpoint	Designed	Running	Who
10.10.10.21-23:4222 client	yes	yes	middleware, access-*, billing, worm/tree, identity, CS/sales/acct
10.10.10.21-23:6222 cluster routes	yes	yes (private <code>vmbr1</code> only)	nats-a/b/c to each other
10.10.10.21-23:7422 leaf hub	yes (public- edge)	yes	public edge remotes
127.0.0.1:8222 monitor on each LXC	yes	yes	operators on the guest

Endpoint	Designed	Running	Who
70.88.205.138:4222 public client	yes (token)	yes	leaf edge; not Zapier
127.0.0.1:14422 WS on edge	yes	yes	Caddy → wss:// nats.zapier.georgelambert.org
127.0.0.1:8229 edge monitor	yes	yes	host loopback
:6222 / :7422 / :8222 on vmbr0	no	must stay closed	—
Host leftover 127.0.0.1:4222 (docker-proxy) + 10.10.10.1:4222 socat	lab leftover	present	do not point product NATS_URL here
MQTT :1883	study only	removed	—

Internal keep/fleet/middleware URL:

nats://10.10.10.21:4222,nats://10.10.10.22:4222,nats://
10.10.10.23:4222

Public (token in /etc/nats/public-edge.token on NS1): nats://
<token>@70.88.205.138:4222. Isolation on the hub is still \$G + private LAN.
nkeys/mTLS sketched in verae-nats-accounts, **not** flipped.

JetStream streams (running)

Created by verae-nats-cluster/scripts/ensure-streams.sh (middleware
ensureStreams also creates the first three).

Stream	Subjects	Replicas	Storage	Who consumes
ZAPIER_JOBS	verae.zapier.jobs.watch	3	file	job-poller (in middleware process), Q job-poller middleware wait + webhook router (event- webhook-
ZAPIER_EVENTS	verae.zapier.jobs.events	3	file	

Stream	Subjects	Replicas	Storage	Who consumes
ZAPIER_WEBHOOKS	verae.zapier.webhooks.deliver	3	file	router); optional exclusive jobs-events webhook- deliver (in middleware), Q webhook- deliver no durable consumer (USAGE_WRITER named only)
ZAPIER_USAGE	verae.zapier.usage	3	file	worm + tree on LXC 510 (keep); aggregator not fleet- enabled
VERAE_ARCHIVE	verae.archive.>	3	file	

Billing is **core request-reply / pub**, not a JetStream stream.

Address inventory

Address	Kind	Designed	Running	Notes
verae.zapier.jobs.watch	JS work Q job-poller	yes	yes	middleware consumes
verae.zapier.jobs.events	JS events	yes	yes	poller publishes worker routes
verae.zapier.webhooks.deliver	JS work Q webhook-deliver	yes	yes	same Node p
verae.zapier.usage	JS optional	yes	stream yes , consumer no	middleware
verae.internal.jobs.events	core republish	yes (jobs-events exclusive)	partial	only if JOBS_

Address	Kind	Designed	Running	Notes
verae.billing.statement.get	RR Q account-balance	yes	yes	portal, CS, s
verae.billing.balance.get	RR	yes	yes	account-bala
verae.billing.balance.adjust	RR	yes	yes	CS credits, p
verae.billing.usage.recorded	pub	yes	yes if edge NATS _URL set	zappier/src account-bala
verae.billing.payment.recorded	pub	yes	yes if NATS _URL	portal reload
verae.billing.credit.applied	pub	yes	yes if NATS _URL	admin/CS (a
verae.billing.customer.put	pub	yes	yes if NATS _URL	ledger.ts; a
verae.access.authz.check	RR Q access-authz	yes	yes	fleet keepFlo
verae.access.authz.deny	pub audit	yes	yes (on deny)	no SIEM sinl
verae.access.web.in / verae.access.web.billing.*	ingress	yes	yes	mapped after e.g. verae.a
verae.access.api.*	ingress	yes	yes	
verae.access.zapier.*	ingress	yes	yes	HTTPS plan
verae.access.leaf.in / verae.access.leaf.*	ingress	yes	yes	no billing
verae.access.staff.*	ingress	yes	yes	CS/sales/acc
verae.identity.bind	RR Q identity	yes	yes	fleet
verae.identity.lookup	RR Q identity	yes	yes	
verae.archive.put	JS (VERAE_ARCHIVE)	yes	partial	splitter in-p on 510 cons
verae.archive.query	broadcast (no Q)	yes	partial	live aggregat (simulator h
verae.archive.reply.<id>	replies on bloom hit	yes	partial	worm/tree o fleet-enable
verae.splitter.in	future	yes (docs)	no	splitter stay
verae.example.process.in / .out / .template	template	yes	no	verae-nats-
bench.*	lab only	n/a	ephemeral	speed studie

Plane mapping (verae-access-authz): internal
 verae.billing.statement.get + plane web → ingress
 verae.access.web.billing.statement.get. Planes never publish internal
 subjects until authz.check returns allow.

Durable consumers (named in middleware)

Name	Stream / subject	Running
job-poller	ZAPIER_JOBS / jobs.watch	yes (in-process)
event-webhook-router	ZAPIER_EVENTS / jobs.events	yes (in-process)
webhook-deliver	ZAPIER_WEBHOOKS / webhooks.deliver	yes (in-process)
usage-writer	ZAPIER_USAGE / verae.zapier.usage	no
jobs-events-exclusive	ZAPIER_EVENTS	no unless JOBS_EVENTS_EXCLUSIVE=1

Fleet overlay vs in-process

packages/verae-fleet/overlays/ns1/fleet.json (2026-09-14):

Service	fleet enabled	Actual lab
middleware-http	true	running ; starts job-poller + webhook workers in-process when NATS_ENABLED
job-poller	false	covered by middleware; do not spawn a second copy
webhook-deliver	false	same
archive-aggregator	false	not running as a fleet worker
archive-worm / tree-node	false on host fleet	running on LXC 510 via keep units.px-worker.json (1 worm + 1 tree)

Service	fleet enabled	Actual lab
account- balance, access-*, identity, jobs- events, CS/ sales/acct, staff-iam/ session, zappier-edge	true keepFloor	keepFloor
zapier- simulator	true, keepFloor false	optional
nats	managed false	3-node cluster + leftover host loopback

Endpoint I/O

Each product NATS address: **what it is for, who sends, who receives, how to attach.**

Connect with a **long-lived** client (reconnect-per-message is the anti-pattern):

NATS_URL=nats://10.10.10.21:4222,nats://10.10.10.22:4222,nats://10.10.10.23:4222

Public edge (token in /etc/nats/public-edge.token on NS1): nats://<token>@70.88.205.138:4222. Do not put that URL in Zapier or in Verae.

verae.zapier.jobs.watch

For	Work queue: “poll Verae until this jobId is terminal.”
In from	middleware enqueueWatch after POST /api/timestamp (or mock) returns jobId
Out to	job-poller worker (queue job-poller, durable on ZAPIER_JOBS)
Then	

poller calls Verae GET /api/status/{jobId} over **HTTPS**, not NATS

Payload: tenantId, veraeUserId, jobId, tokenRef, enqueuedAt, attempt, maxAttempts, intervalMs, traceId. JWT stays in middleware (tokenRef); never on Zapier.

Connect Verae-side: do **not** subscribe this from api.veraetime.net. The poller is the Verae HTTP client. To add another watcher, start another job-poller consumer in the same queue group.

verae.zapier.jobs.events

For	Terminal outcomes: timestamp.completed / failed / timeout
In from	job-poller after Verae status is not pending
Out to	(1) middleware waitForJobEvent (wait HTTP); (2) webhook router → webhooks.deliver; (3) optional verae-jobs-events mailbox :3030

If JOBS_EVENTS_EXCLUSIVE=1, only verae-jobs-events is the durable consumer and republishes verae.internal.jobs.events. Default lab: middleware wait + webhook worker share the stream.

Payload: event, tenantId, jobId, status, traceId, emittedAt.

verae.zapier.webhooks.deliver

For	HTTPS POST to hooks.zapier.com (REST Hook)
In from	webhook worker on completed/failed events
Out to	Zapier cloud HTTPS , never NATS

Payload: hookId, tenantId, targetUrl, event, payload, attempt, traceId.
Slow consumer: keep replica floors on the worker if you split it out of
middleware.

verae.zapier.usage

For	Optional meter of Zap actions on NATS (JetStream)
In from	designed: middleware / edge
Out to	designed: USAGE_WRITER → books
Running	stream exists; no writer; middleware uses local usage files

Prepaid meter that **is** wired uses `verae.billing.usage.recorded` (core pub), not this stream. Finish: either publish here from `recordUsage` or drop the stream.

Billing (verae.billing.*) — account-balance source of truth

Request-reply unless noted. Queue account-balance. **Not** JetStream.

Address	For	In from	Out to
<code>statement.get</code>	prepaid, credits, usage, payments	portal, CS, sales, accounting, access-web/api	reply to requester
<code>balance.get</code>	cents only	internal	reply
<code>balance.adjust</code>	CS credit or portal reload	CS, access-web (reload kind only)	books + reply
<code>usage.recorded</code>	meter pub	zappier-edge meter() onUsage when NATS_URL set	account-balance subscriber (debits prepaid)
<code>payment.recorded</code>	reload/ invoice pub	portal	books
<code>credit.applied</code>	goodwill pub	CS/admin	books
<code>customer.put</code>	upsert customer	edge ledger	books

Authz **must** pass `verae.access.authz.check` before these from a public plane. Leaf plane **cannot** call billing (policy).

Middleware `recordUsage` still writes **local tenant files** and does **not** publish `usage.recorded`. Two meters until that is unified.

Access planes

Address	For	In	Out
<code>verae.access.authz.check</code>	allow/deny { plane, subject, principal }	every plane + edge billing- nats.ts	{ allow, reason }; deny also pubs <code>authz.deny</code>
<code>verae.access.authz.deny</code>	audit	authz on deny	nowhere yet (no SIEM)
<code>verae.access.web.in /</code> <code>verae.access.web.billing.*</code>	browser portal	access- web HTTP	<code>statement.get /</code> <code>balance.adjust</code> <code>kind=reload</code> after authz
<code>verae.access.api.*</code>	customer x- api-key	access- api	<code>statement;</code> HTTPS to middleware for jobs jobs.watch via HTTP to middleware, not Zapier NATS; <code>usage.recorded</code> allowed by policy
<code>verae.access.zapier.*</code>	Zapier HTTPS only	access- zapier	archive + jobs after authz; never billing
<code>verae.access.leaf.in</code>	S2S { token, target, payload }	remote leaf	archive + jobs after authz; never billing
<code>verae.access.staff.*</code>	CS/sales/ accounting	access- staff	billing after authz

Ingress mapping: `accessAddress(plane,`
`'verae.billing.statement.get') →`
`verae.access.<plane>.billing.statement.get.`

Identity

Address	For	In	Out
verae.identity.bind	bind email/ customerId → veraeUserId	portal / middleware	reply
verae.identity.lookup	lookup by customerId or veraeUserId	same	reply

Queue identity.Fleet keepFloor.

Archive / tree

Address	For	In	Out
verae.archive.put	WORM/tree write (kind: publicMeta / privateMeta / file / tree)	splitter (in- process from middleware today)	every worm/tree (JS)
verae.archive.query	broadcast lookup	aggregator	all worm/ tree, no queue group
verae.archive.reply.<correlationId>	bloom hit only	worm/tree	aggregator

Silence = miss. Chain seal does not wait on archive. Bytes never go to the chain.

Template / unused

Address	For	In	Out	Running
verae.splitter.in	future split-out splitter process	middleware	splitter	no

Address	For	In	Out	Running
verae.example.process.in	template worker	any	example-process Q	no
verae.example.process.out	template fan-out	template	peers	no
verae.example.process.reply.<id>	template reply	template	requester	no
verae.internal.jobs.events	exclusive mailbox republish	jobs-events	wait/webhook	no unless env

How to connect Verae API to NATS

api.veraetime.net is **HTTPS**. It does not subscribe. The glue is middleware:

1. Set `VERAE_API_BASE_URL=https://api.veraetime.net` and `MOCK_VERAE=false`.
2. Middleware login (username/password) → JWT stored as tokenRef.
3. `POST /api/timestamp` → `202 { jobId }`.
4. Publish `verae.zapier.jobs.watch`.
5. Poller `GET /api/status/{jobId}` until completed / failed.
6. Publish `verae.zapier.jobs.events`.
7. Wait HTTP and/or REST Hook.

Do **not** point Verae at `NATS_URL`. Do **not** put NATS in Zapier env. A new worker that must see seals uses `jobs.events` or `HTTP GET /zapier/v1/status/{jobId}`.

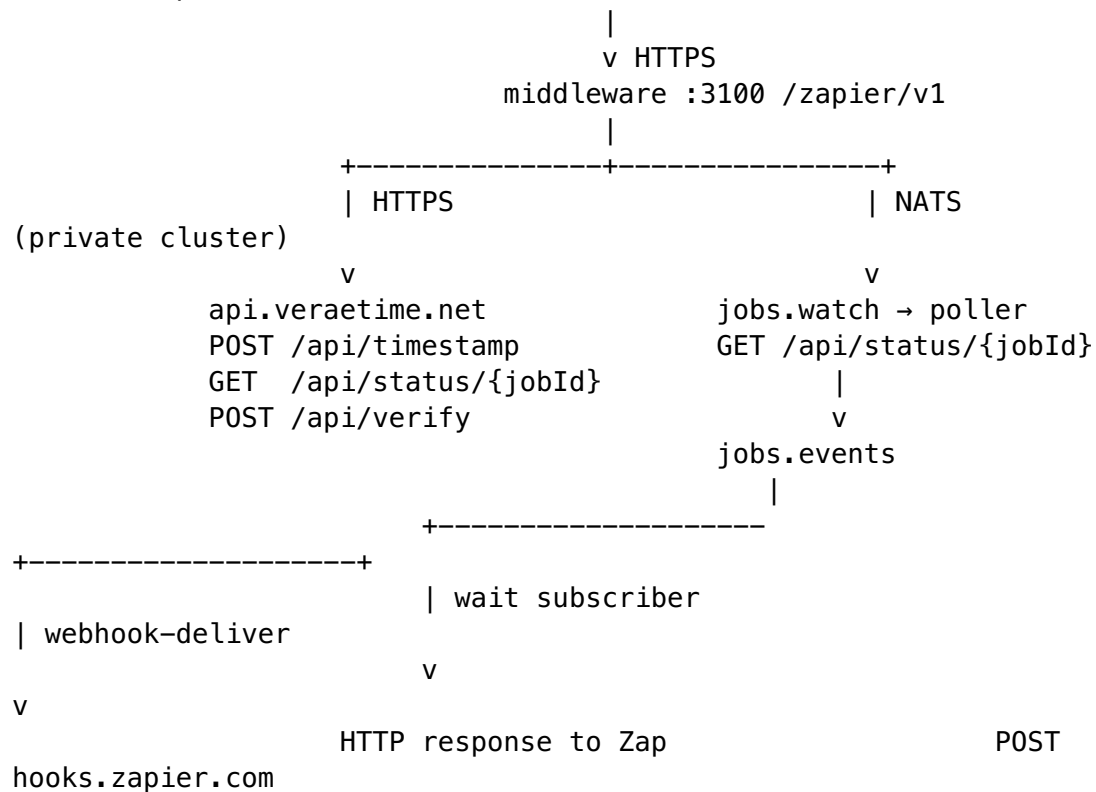
Template for a new addressed process: clone `verae-nats-process`, rename `verae.example.process.*`, add the subject to `verae-access-authz` policy for the plane that may call it.

Zapier → Verae API → Zapier

How a timestamp job is submitted, watched, and returned. Lab today: `MOCK_VERAE=true`. Live path is the same HTTP contract with `MOCK_VERAE=false`.

Rule

Zapier cloud --HTTPS only--> zap.zapier.georgelambert.org
(access-zapier)



Zapier never has NATS_URL. Verae never has NATS_URL.

1. Submit (async create)

1. Zap action **Create Timestamp** (or curl) POST https://zap.zapier.georgelambert.org/zapier/v1/timestamp with portal API key zmw_... (not a Verae JWT).
2. access-zapier / edge: authz, then meter() records cents; if NATS_URL is set, natsPublish(verae.billing.usage.recorded).
3. Forwards to middleware POST /zapier/v1/timestamp.
4. Middleware checkEntitlement (local plan limits today).
5. veraeClient.createTimestamp:
 - **Mock:** in-process job, fake jobId.
 - **Live:** POST https://api.veraetime.net/api/timestamp with JWT → 202 { jobId }.
6. Splitter (in-process): hash vs attachments; optional verae.archive.put (bytes never go to chain).
7. enqueueWatch → JetStream **verae.zapier.jobs.watch**.
8. HTTP 202 { jobId, sha256, existing } back to Zapier immediately.

Same SHA-256 is idempotent (existing: true).

2. Monitor (job poller)

The poller is **in the middleware Node process** when `NATS_ENABLED=true` (`startJobPollerWorker`). Fleet does **not** spawn a second copy (`job-poller.enabled: false`).

1. Durable consumer `job-poller` on `ZAPIER_JOBS / verae.zapier.jobs.watch`.
2. Resolve `tokenRef` → tenant Verae password → POST login → JWT (stays in middleware).
3. GET `/api/status/{jobId}` (mock or live timestamp server).
4. pending → nak after `intervalMs` (retry).
5. completed → publish `jobs.events { event: timestamp.completed, status, jobId, tenantId, traceId }`.
6. failed → `timestamp.failed`. Too many attempts → `timestamp.timeout`.
7. ack the watch message.

Live OpenAPI: poll until completed / failed. Hash lookup is **501** on live (`lookupHash`). Map `certificate`, `timestamp`, `blockIndex`, `sha256` into the receipt Zap samples expect.

3. Wait (sync Zap step)

Create Timestamp and Wait → POST `/zapier/v1/timestamp/wait`.

1. Same as submit (watch is already on NATS).
2. `waitForJobEvent(jobId, WAIT_TIMEOUT_MS)` subscribes to **`verae.zapier.jobs.events`**.
3. On matching `jobId` event: return status JSON (certificate / receipt).
4. On timeout: `{ status: pending, jobId }` so the REST Hook can finish the Zap.

`includeAttached`: after seal, aggregator should `verae.archive.query`; bloom hits merge extra receipts. **Simulator does this; live aggregator hook is TBD.**

4. Push back to Zapier

- **Wait response**: HTTP body on the same Zap step.
- **REST Hook**: Zapier subscribes POST `/zapier/v1/hooks`. On `jobs.events`, middleware enqueues **`verae.zapier.webhooks.deliver`**. Worker POSTs `targetUrl` (`hooks.zapier.com`).

- **Poll status:** Zap action **Find Job Status** → GET /zapier/v1/status/{jobId} → middleware GET Verae (or last event).

5. Verify

POST /zapier/v1/verify → middleware → Verae POST /api/verify. No job watch.

6. Batch Merkle (designed; same bus)

From MESSAGE-FLOWS.md:

1. POST /v1/timestamp/batch — one item per line.
2. Middleware hashes leaves, builds Merkle tree.
3. Chain seals **only the root** (POST /api/timestamp on the root).
4. Each leaf proof → verae.archive.put kind: tree.
5. Central hash lookup of a leaf misses unless includeTree=true.
6. Aggregator broadcasts verae.archive.query kinds=["tree"].
7. Tree node bloom hit → archive.reply.<id> with proof.

Live wait path does not yet call the aggregator, so tree-leaf receipts are not merged on NS1.

Connect live api.veraetime.net

Still blocked on operator bind credentials. When ready:

```
# middleware env
MOCK_VERAЕ=false
VERAE_API_BASE_URL=https://api.veraetime.net
NATS_ENABLED=true
NATS_URL=nats://10.10.10.21:4222,nats://10.10.10.22:4222,nats://10.10.10.23:4222
JETSTREAM_REPLICAS=3
```

Then: working Verae login, one live seal, confirm 202/status/verify, map receipt fields, REST Hook to a private Zap. See zapier-decisions/STATUS.md “Once MOCK_VERAЕ=false”.

Billing on this path

Hop	Designed	Running
Edge meter	verae.billing.usage.recorded	

Hop	Designed	Running
		coded ; fires when edge has NATS_URL
Account-balance	subscribe + debit prepaid; 402 if empty	subscriber running ; 402 from books not proven on every Zap
Middleware quota	local files via recordUsage	running ; not NATS
verae.zapier.usage	optional JS meter	stream only

Connect a new internal process

```
export NATS_URL=nats://10.10.10.21:4222,nats://
10.10.10.22:4222,nats://10.10.10.23:4222
# clone verae-nats-process, rename subjects, register in verae-
fleet
```

Jobs stay: HTTP in from Zapier → HTTPS to Verae → NATS watch/events → HTTP back to Zapier.

Gaps and finish list

Modules and addresses that are designed but **not fully connected, not running as fleet workers, or not on the live Verae/Zapier path.**

Blocked on the operator (not a NATS bug)

Item	State	To finish
Zapier Platform push	no login	zapier login → register → MIDDLEWARE_BASE_URL=https://zap.zapier.georgelambert.org → push private app
Live Verae	MOCK_VERAE=true	credentials; MOCK_VERAE=false; prove one seal
nkeys / mTLS	cluster \$G only	verae-nats-accounts sketch; flip only after every NATS_URL has creds

Item	State	To finish
Hash lookup on live OpenAPI	501	Verae API gap or our store

Running on HTTP/NATS but incomplete routing

Module	Running?	NATS	Gap
middleware HTTP + in- process job- poller + webhook- deliver	yes	watch/events/ deliver	live Verae + Zapier push
account- balance	yes	billing RR + usage.recorded sub	prove 402 from books on a Zap when prepaid is 0
zapper-edge	yes	usage.recorded / payment.recorded / credit.applied / customer.put if NATS_URL	confirm live process env; Stripe live
access-* + authz	yes	authz + ingress	—
identity	yes (fleet)	bind/lookup	confirm portal always binds veraeUserId
jobs-events mailbox :3030	yes (fleet)	events counter	exclusive consumer off unless JOBS_EVENTS_EXCLUSIVE=1
CS / sales / accounting	yes	statement/adjust	staff UX
archive-worm + tree-node	yes on LXC 510 (keep)	archive.put/query/ reply	only 1+1 copies; fleet overlay enabled: false for extra replicas
archive- aggregator	no as fleet worker	query/reply	enable overlay or finish in-process hook (includeAttached TBD in middleware)
request- splitter	library in middleware	archive.put	verae.splitter.in not used

Module	Running?	NATS	Gap
ZAPIER_USAGE / USAGE_WRITER	stream yes	verae.zapier.usage	no consumer; middleware usage is local files
fleet job- poller / webhook- deliver extra copies	overlay false	same subjects	only enable if you want SSH-spread replicas instead of in-process (same queue names)

Designed, not a live worker

Module	Address	What to do
verae-nats- process	verae.example.process.*	template only; clone to add a function
verae-activate	none (HTTPS Zapier)	optional tiny app; needs Zapier login
verae-zapier (full app)	HTTPS	push after login
simulator	in-process fake NATS	local :3847; not production
verae-nats- accounts	nkeys policy	not flipped

Aux (no product subjects, or HTTP only)

Module	NATS	Notes
verae-keep	none	HTTP health; restarts units
verae-fleet	none	catalog + keepFloor
verae-uptime	none	off-box HTTPS GET of public doors
verae-backup	none	disk / JetStream snapshot; no live event subject
verae-bootstrap / verae- deploy / verae-proxmox / verae-ops	none	install/link
staff-iam / staff-session / staff-ui	none	HTTP cookies

Module	NATS	Notes
public NATS edge	leaf, token	do not use from Zapier; rotate token; later nkeys LEAF vs INTERNAL

Billing triggers and event logging

Signal	Designed	Running	Finish
Timestamp/ verify meter (edge)	verae.billing.usage.recorded	code path exists (zappier/ src/ app.ts onUsage)	Confirm NATS_URL on live edge; 402 from account- balance when prepaid empty
Middleware quota	local files recordUsage	yes	Either publish usage.recorded too or drop duplicate quota
Optional JS meter	verae.zapier.usage + USAGE_WRITER	stream only	Wire or delete
Portal reload	payment.recorded + balance.adjust	coded	confirm both fire on Stripe/ test reload
CS credit	balance.adjust + credit.applied	coded	confirm applied pub on live
Customer upsert	customer.put	coded	—
Authz deny audit	verae.access.authz.deny	running on deny	no SIEM sink
Job lifecycle log	jobs.events	running	mailbox counts only — add structured log/ export if needed
Archive put/query	VERAE_ARCHIVE	510 worm/ tree	aggregator not in wait path
Public edge	leaf	running	

Signal	Designed	Running	Finish
			rotate token; do not expose :6222

Ordered finish for message routing

1. **Do not double-start** job-poller: either in-process **or** fleet copies, same queue job-poller.
2. Confirm edge **NATS_URL** so usage.recorded actually lands; then 402 from books. Unify or drop middleware local recordUsage.
3. Enable **archive-aggregator** (or in-process equivalent) so includeAttached and tree proofs work on wait/status.
4. Raise worm/tree **replica floors** on 510/more CTs if archive HA is required (fleet overlay min 3).
5. Operator: Zapier push + **MOCK_VERAЕ=false**.
6. Prove: Zap create → jobs.watch → live GET /api/status → jobs.events → wait body + REST Hook.
7. nkeys/mTLS after every client has creds; then lock down the public edge to accounts (LEAF vs INTERNAL).
8. Optional: exclusive jobs-events mailbox; SIEM on authz.deny and jobs.events; verae.splitter.in if splitter is split out; USAGE_WRITER or delete ZAPIER_USAGE.

What is already enough for a mock Zap

HTTPS zap/mw → mock create → jobs.watch → in-process poller → mock status completed → jobs.events → wait returns mock certificate. Archive extras and live chain are the missing middle.

Developer credentials and connection info

Lab pack for the Verae Time × Zapier stack on **NS1.GEORGELAMBERT.ORG** (70.88.205.138). Values below were checked **2026-09-14**. These are **lab defaults**, not production secrets. Rotate the public NATS token and admin secrets before treating this as a customer environment.

Zapier cloud **never** gets a NATS URL or a Verae password.

1. Who you are talking to

Host	Role
70.88.205.138	NS1 Proxmox. Public HTTPS + public NATS client edge
10.10.10.20	LXC 510 verae-px-worker (worm + tree)
10.10.10.21-23	LXC 511-513 nats-a/b/c JetStream cluster verae

SSH to the box: `ssh marchon@70.88.205.138`

Git SSH: `ssh://git@git.georgelambert.org:2223/marchon/<repo>.git`
(port 2223)

2. Public HTTPS doors (live)

Wildcard `*.zapier.georgelambert.org` → 70.88.205.138.

URL	Process	Port	Last check
<code>https://zapier.georgelambert.org/</code>	docs catalog	files	200
<code>https://git.georgelambert.org/</code>	Forgejo	:443 / SSH :2223	200
<code>https://portal.zapier.georgelambert.org/</code>	access-web → /portal/	:3021	302 → portal
<code>https://api.zapier.georgelambert.org/health</code>	access-api	:3022	200
<code>https://leaf.zapier.georgelambert.org/health</code>	access-leaf	:3023	200
<code>https://zap.zapier.georgelambert.org/health</code>	access-zapier	:3024	200
<code>https://staff.zapier.georgelambert.org/</code>	access-staff	:3025	302 login
<code>https://login.zapier.georgelambert.org/</code>	staff-session	:3027	200
	staff-iam	:3028	200 (users: 5)

URL	Process	Port	Last check
https:// iam.zapier.georgelambert.org/ health			
https:// cs.zapier.georgelambert.org/	customer- service	:3011	302
https:// sales.zapier.georgelambert.org/	sales- pricing	:3012	302
https:// acct.zapier.georgelambert.org/	accounting- export	:3013	302
https:// mw.zapier.georgelambert.org/ health	middleware	:3100	200 mockVerae:true natsConnected:true
https:// fleet.zapier.georgelambert.org/	fleet console	:3850	302
https:// nats.zapier.georgelambert.org/	NATS WebSocket (Caddy)	14422	400 on GET (expected; it is WS)

Loopback on NS1 (not public): zapier-edge **127.0.0.1:13000** (admin + Stripe + SQLite). Tunnel:

```
ssh -L 13000:127.0.0.1:13000 marchon@70.88.205.138
# then http://127.0.0.1:13000/admin and /docs
```

Operator console tunnel: ssh -L 3850:127.0.0.1:3850
marchon@70.88.205.138

3. NATS

Internal cluster (workers, keep, fleet)

No token. Private vmbr1 only.

```
NATS_URL=nats://10.10.10.21:4222,nats://10.10.10.22:4222,nats://
10.10.10.23:4222
JETSTREAM_REPLICAS=3
```

Port	Bind	Use
:4222	10.10.10.21-23	clients
:6222	same, vmbr1	cluster routes — do not publish
:7422	same	leaf hubs for the public edge — do not publish

Port	Bind	Use
:8222	127.0.0.1 on each LXC	monitor

From a laptop you **cannot** reach 10.10.10.x unless you are on NS1 or you tunnel:

```
ssh -L 14222:10.10.10.21:4222 -L 14223:10.10.10.22:4222 -L
      14224:10.10.10.23:4222 marchon@70.88.205.138
export NATS_URL=nats://127.0.0.1:14222,nats://
      127.0.0.1:14223,nats://127.0.0.1:14224
```

Public client edge (token)

Live token (file /etc/nats/public-edge.token on NS1):

```
4743a49383ce2ce86e99d9d41c45c0004fe75eb23ef3a2b9
```

```
nats://
4743a49383ce2ce86e99d9d41c45c0004fe75eb23ef3a2b9@70.88.205.138:4222
wss://nats.zapier.georgelambert.org
```

CONNECT with that token returns PONG (checked 2026-09-14).
auth_required: true. Server name nats-public-edge. **Do not** put this URL in Zapier. Recreate: bash packages/verae-nats-cluster/scripts/expose-public.sh on NS1.

Leftover lab listeners (ignore for new clients): host 127.0.0.1:4222 docker-proxy, 10.10.10.1:4222 socat. zappier-edge process still has NATS_URL=nats://127.0.0.1:4222 — billing pubs may miss the 3-node cluster until that env is flipped.

nkeys/mTLS: **not** on. Isolation is LAN + token.

4. Middleware (jobs)

Live: MOCK_VERAЕ=true, NATS_ENABLED=true, cluster URL as above, ADMIN_SECRET=change-me-admin (default), TOKEN_SECRET=dev-secret-change-me (default).

Lab tenant (created 2026-09-14 for this pack)

Field	Value
tenantId	tenant-dev-lab-local-89ab82de
plan	free

5. Portal / zappier-edge (billing users)

SQLite /SSD3/moved/home-marchon/verae-stack/packages/zappier/
zappier.db. Bind 127.0.0.1:13000.

Seed **customer API keys** (x-api-key on edge /v1/*, **not** middleware zmw_):

Customer	tier	key
Ada (free)	free	key-ada
Grace (pro)	pro	key-grace
Linus (business)	business	key-linus
Docs Capture	free	key-021851970d8a22c6

These keys are for the **edge meter / portal**. Middleware jobs need a zmw_... key (section 4).

Seed **admin console** (/admin, x-admin-key or login):

username	password
admin	admin-dev-key
demo	\$\$\$Adm1n###

ZAPPIER_ADMIN_KEY default on CS/sales/staff processes: admin-dev-key.

Stripe: **not** live (lab quotes only). Env STRIPE_* unset on the sampled edge process.

Signup more customers at <https://portal.zapier.georgelambert.org/portal/>

6. Staff IAM

<https://login.zapier.georgelambert.org/> · <https://iam.zapier.georgelambert.org/>

Seed users (defaults; live IAM has these five):

username	password	role
admin	admin-dev-key	owner
cs	cs-dev-key	cs
sales	sales-dev-key	sales
accounting	acct-dev-key	accounting
operator	fleet-dev-key	operator

STAFF_KEY / session fallback: admin-dev-key.

7. Leaf plane

<https://leaf.zapier.georgelambert.org/health> → `verae.access.leaf.in`

Live: `LEAF_TOKEN=leaf-dev-token`

```
curl -sS -H "content-type: application/json" \
  -d '{"token":"leaf-dev-token","target":"verae.archive.put","payload":
    {"principal":"dev"}}' \
  https://leaf.zapier.georgelambert.org/
```

Billing targets are **denied** on this plane.

8. Git (Forgejo)

Web	https://git.georgelambert.org/marchon/
SSH	<code>ssh://git@git.georgelambert.org:2223/marchon/<name>.git</code>
User	<code>marchon</code>
Overview of routes	https://git.georgelambert.org/marchon/dev-overview
Monorepo	<code>master-zapier-plan-draft</code>
Bootstrap	https://git.georgelambert.org/marchon/verae-bootstrap

```
GIT_SSH_COMMAND="ssh -p 2223" git clone ssh://git@git.georgelambert.org:2223/marchon/dev-overview.git
```

Add your SSH public key on Forgejo (user **marchon** or a new account). Do not commit deploy tokens.

Clone-all:

```
git clone ssh://git@git.georgelambert.org:2223/marchon/verae-bootstrap.git
cd verae-bootstrap
export VERAЕ_SRC=$HOME/verae-src
bash scripts/fetch.sh ns1-all-in-one
```

9. Environment cheat sheet (new worker on the cluster)

```
export NATS_URL=nats://10.10.10.21:4222,nats://10.10.10.22:4222,nats://10.10.10.23:4222
export JETSTREAM_REPLICAS=3
export NATS_ENABLED=true
export MOCK_VERAЕ=true           # flip to false only with live Verae login
export VERAЕ_API_BASE_URL=https://api.veraetime.net
export AUTHZ_URL=http://127.0.0.1:3020
export ZAPPIER_UPSTREAM=http://127.0.0.1:3100
export ZAPPIER_ADMIN_URL=http://127.0.0.1:13000
export ZAPPIER_ADMIN_KEY=admin-dev-key
export LEAF_TOKEN=leaf-dev-token
export STAFF_IAM_URL=http://127.0.0.1:3028
```

Template worker: clone verae-nats-process, rename verae.example.process.*.

10. Not issued yet (blocked)

Item	Why
Zapier Platform deploy key / zapier login	operator has not logged the CLI in
Live api.veraetime.net username/ password	MOCK_VERAЕ=true; JWT would stay in middleware
NATS nkeys / account creds	cluster is \$G + LAN
Stripe live secret	lab only
lan-134 SSH worker	disabled

When Zapier login exists: MIDDLEWARE_BASE_URL=https://zap.zapier.georgelambert.org, auth field api_key = zmw_....

When live Verae exists: middleware MOCK_VERAЕ=false + VERAЕ_API_BASE_URL=https://api.veraetime.net + working login. Zapier still never sees that password.

11. Related docs

- Routes: [01-nats-routes.md](#)
- Endpoint I/O: [02-endpoint-io.md](#)
- Job path: [03-zapier-verae-path.md](#)
- Gaps: [04-gaps-and-finish.md](#)

- Ops start: <https://zapier.georgelambert.org/packages/verae-ops/GETTING-STARTED.pdf>
- Status: <https://zapier.georgelambert.org/packages/zapier-decisions/STATUS.pdf>