Glitch RunMyCRM — Beta Launch Playbook

Updated: 2026-05-04 EOD **Goal:** Ship the first 1-2 beta testers TODAY without waiting for marketplace approval. Manual fallback for steps that aren't fully automated yet.

---

1. The 2 active payment links (only what beta needs)

Tier	Price	URL	Product (Stripe)	tier metadata	Path
Employee	$99/mo	https://link.runmycrm.com/payment-link/69f3c8e4b615f70a8a33aad6	`prod_UQmQirMf9RNRZ4`	`tier-employee`	GHL → GHL workflow → WF13
Glitch Beta	$197/mo	https://buy.stripe.com/8x2cN54KSf1k1i1eqm9k40g	`prod_UQWAjfR9B8PaEn`	`tier-beta-tester`	Stripe direct → Stripe webhook → WF4/Feeder

Skip Starter / Pro / Agency for now — products + prices exist in Stripe with metadata, but no payment link until owner needs them.

---

2. The provisioning chain (what fires when payment hits)

Beta path (Stripe direct)

```
Customer pays at buy.stripe.com/8x2cN54KSf1k1i1eqm9k40g
  → Stripe processes → fires events to BOTH webhooks:
       /webhook/stripe-event        → JTWWQKIG1YoHcVSE Stripe Subscription Feeder (active)
       /webhook/stripe-glitch       → ABgoe6L5acOMZPHm WF4 Stripe Handler (active)
  → Feeder waits 20s, looks up location by stripe customer, forwards to provisioner
  → Provisioner mints OpenRouter child key, pushes snapshot, runs provision.sh
```

Employee path (GHL link)

```
Customer pays at link.runmycrm.com/...
  → GHL processes → fires GHL workflow on payment success
  → GHL workflow POSTs to WF13 Beta Payment Provisioner webhook (yl6z6lSZzD7zQ9fK)
  → WF13 sends SMS + email + tags + Telegram, prepares onboarding
```

Both webhooks are active and listening. Two questions only real payment will answer: - Does Stripe webhook signature verification work? (Stripe webhook signing secret must be in n8n vars) - Does the lookup-by-customer step find the right tenant? (assumes customer email matches)

---

3. Manual onboarding fallback for first 1-2 testers (10 min each)

Run this if the auto chain breaks. Or run it for the FIRST tester anyway to verify each step works in isolation, then trust the chain for #2.

Step A — Customer pays

Wait for Stripe/GHL payment confirmation. Capture: email, name, business name, phone.

Step B — Create their GHL sub-account

GHL agency UI: 1. Sub-Accounts → + Add Sub-Account 2. Name: `<Business Name>` 3. Owner email: `<their email>` 4. Snapshot: select **"RunMyCRM Glitch v1"** (id `hyGfwGk0Elq0OGLZlXxU`) 5. Save → wait ~30s

Note the new `locationId`.

Step C — Generate sub-account PIT

1. Switch into the new sub-account
2. Settings → Integrations → Private Integrations
3. Create PIT named `Glitch BusyBee` with scopes: - contacts.readonly + write, opportunities.readonly + write - calendars.readonly + write, conversations/message.readonly + write - locations.readonly, locations/customValues.readonly + write - users.readonly
4. Copy the PIT (`pit-...`)

Step D — Mint per-tenant OpenRouter child key + write to GHL

SSH to droplet:

```
ssh root@142.93.60.203
/opt/claudebotz/venv/bin/python3 - <<'PY'
import os, json, urllib.request
from dotenv import load_dotenv
load_dotenv("/opt/claudebotz/.env")

LOCATION_ID = "<paste new locationId>"
TENANT_PIT = "<paste new sub-account PIT>"
LABEL = "<business name>"
LIMIT_USD = 10  # Beta tier — adjust per plan

# Mint child key
mgmt_key = os.environ["OPENROUTER_MGMT_KEY"]
req = urllib.request.Request(
    "https://openrouter.ai/api/v1/keys",
    data=json.dumps({"name": f"glitch-{LOCATION_ID}", "label": LABEL, "limit": LIMIT_USD}).enc
    headers={"Authorization": f"Bearer {mgmt_key}", "Content-Type": "application/json"},
    method="POST",
)
res = json.loads(urllib.request.urlopen(req).read())
child_key = res.get("key")
print(f"Child key: {child_key}")

# Write to GHL custom value openrouterapikey
ghl_req = urllib.request.Request(
    f"https://services.leadconnectorhq.com/locations/{LOCATION_ID}/customValues",
    data=json.dumps({"name": "openrouter_api_key", "value": child_key}).encode(),
    headers={"Authorization": f"Bearer {TENANT_PIT}", "Version": "2021-07-28", "Content-Type"
    method="POST",
)
print("GHL response:", urllib.request.urlopen(ghl_req).read().decode())
PY
```

Step E — Spin up tenant BusyBee on droplet

```
/opt/glitch/busybee/provision.sh add <locationId> <PIT>
pm2 save
```

This: allocates next port (9101+), sources `.env-shared` (incl GHL refresh token), spawns PM2 process `ghl-<locationId>`, drops nginx config, reloads nginx → tenant BusyBee live at `https://agents.realtalkha.com/ghl/<locationId>/`.

Step F — Tag the contact properly

In GHL sub-account UI, manually add the right tag: - Employee subscriber → `tier-employee` - Beta subscriber → `tier-beta-tester` + `glitch-onboarding-start`

This unblocks the existing tag-driven workflows (WF12 web scrub, etc.).

Step G — Send welcome

- Their GHL login URL
- Their Telegram bot username (start with master `@GlitchSuperBot` for first beta; per-tenant bot later)
- "Reply to this message to test Glitch"

---

4. After 1-2 successful manual onboards → trust the chain

Once you've validated each step manually, the auto-chain can take over for tester #3+: - Stripe webhooks already fire to n8n - GHL agency-level LocationCreate webhook needs to be configured to point at `https://realtalkha.app.n8n.cloud/webhook/ghl-subaccount-created` (one-time, in GHL agency UI) - WF13 + Feeder + WF4 + EU9jr1lVoSP05T9C (provisioner) chain handles it from there

If something breaks, n8n executions tab shows exactly which node + why.

---

5. Risks to watch on first beta payment

Risk	Mitigation
Stripe webhook signature fails	Confirm `STRIPE_WEBHOOK_SECRET` is in n8n vars; check JTWWQKIG1YoHcVSE "Verify Stripe Webhook Signature" execution
Provisioner can't find location by customer email	Falls back to manual playbook

Risk	Mitigation
Snapshot push regenerates IDs (QUIRK-005/006)	After snapshot, manually verify tags/fields exist before tagging contact
OpenRouter child key creation 401/403	Ensure `OPENROUTER_MGMT_KEY` is the correct management-scoped key (it is)
`glitch-onboarding-start` tag not yet applied → web scrub doesn't fire	Manually tag in step F

---

6. Skip-for-beta list (don't let any of these block first $)

- □ GHL Marketplace submission
- □ OAuth 2.0 backend
- □ Custom Channel Provider for Telegram
- □ Public landing page
- □ Privacy/ToS/Refund/SLA/DPA (have basic versions ready for due diligence questions, not for beta gating)
- □ Per-tenant brand voice for AI Review Responder

After first revenue, prioritize: ToS+Privacy → marketplace listing → OAuth.

⚠ REQUIRED before first Beta payment — n8n vars

Without these, the Stripe webhook signature verification will throw `SECRET_MISSING` and provisioning will not auto-fire. (Customer payment still succeeds; you'd just fall back to manual playbook §3.)

n8n cloud Settings → Variables → Add Variable

Key	Value	Why
`STRIPE_WEBHOOK_SECRET`	`whsec_rdaL87vYYl9CxNfQV3B08rKiAygN2KI6`	Used by Stripe Subscription Feeder (`JTWWQKIG1YoHcVSE`) to verify webhook signatures. Pulled from `/opt/claudebotz/.env`. If signature fails, get the per-endpoint secret from Stripe Dashboard → Developers → Webhooks → `/webhook/stripe-event` → Reveal signing secret.
`OPENROUTER_API_KEY`	already set □	(Set today during AI Review Responder fix.)
`TELEGRAM_BOT_TOKEN`	already set □	
`TELEGRAM_OWNER_CHAT_ID`	already set □	

How to verify it took

After saving, send Heather a test event:

```
# from droplet:
ssh root@142.93.60.203
curl -X POST -H "Stripe-Signature: t=$(date +%s),v1=fake"      -H "Content-Type: application/
```

The response should be `SIGNATURE_MISMATCH` (not `SECRET_MISSING`). That confirms the secret is loaded — the mismatch is expected because we used a fake signature. Real Stripe-signed payloads will pass.

7. Today's hot seats

Owner	Action	Time
Heather	Send Beta link to first tester	5 min
Heather	When payment fires, watch n8n executions to see what's working / breaking	live
Heather (if auto chain breaks)	Run manual playbook §3 above	10 min
Heather (after first beta is live)	Verify Glitch responds via Telegram for that tenant	5 min

---

2026-05-05 Update — Tenant Chat Live

The chat web URL has been finalized. **Replace any reference to the old `https://glitch-chat.vercel.app/{{location.id}}` URL with the new permanent URL.**

New tenant welcome message (Step G)

Send the new tenant: - Their **GHL login URL** (sub-account dashboard, unchanged) - The **Glitch chat URL**: `https://chat.realtalkha.com/` - Instructions: "Visit the chat URL, enter your email, click the magic link in your inbox. You are now talking to Glitch — your AI operator. Try asking it 'how many contacts do I have?' or 'create a new contact for John Smith'." - Telegram bot username (start with master `@GlitchSuperBot` for first beta if you want a fallback channel)

Snapshot custom menu link (push to all new sub-accounts)

Settings → Company → Custom Menu Links → + Add: - Title: "Chat with Glitch" - URL: `https://chat.realtalkha.com/?email={{user.email}}` - Open in: **New Tab** (iframe mode breaks third-party cookie auth in Chrome) - Show in sidebar: yes

Manual vs automated steps for first 1-2 testers — updated

Step	Status
Stripe payment fires	Auto
GHL sub-account created (SaaS configurator)	Auto
OpenRouter child key minted + written to GHL	Auto
Telegram notification with finalize link	Auto
Owner clicks finalize link, creates GHL PIT named "Glitch BusyBee", pastes, clicks Submit	**Manual ~20 sec**
`provision.sh` spins up BusyBee + nginx	Auto (server-side from finalize form)
Tenant clicks "Chat with Glitch" → magic-link login	Auto

That ~20 sec PIT-paste is the only remaining manual step. Eliminating it requires either GHL OAuth (skipped for beta) or modifying BusyBee to authenticate via the shared v2 JWT in `.env-shared`. Queued as a future enhancement.

Verification checklist additions

After the OpenRouter mint and BusyBee provision, also verify: - [ ] Visit `https://chat.realtalkha.com/`, enter the tenant's email, click the magic link from email or Telegram - [ ] Land on `https://chat.realtalkha.com/<locationId>/` - [ ] Send "what tools do I have?" — should see a `busybee_list_tools` card with a count of 588 (or similar) tools - [ ] Send "find my contacts" — should chain `busybee_list_tools` → `busybee_call(search_contacts)` cards - [ ] Send "make me a logo" — should see a `replicate_run` card with image rendered inline

If any of those fail, check `pm2 logs glitch-chat --err` and the n8n executions tab.