WHOLEREACH · AUTOMATED MARKETING SYSTEM

The Complete User Manual

A self-driving social-media marketing platform: how it works, how to run it, and how to operate it as a managed service for clients.

Version 1.0 · 2026-07-26 · Institutional briefing document
Live edition: wholereach.com/usermanual/

Contents

01

How It Works — Architecture & Flow

The two-server design, the nightly pipeline, and every moving part — from raw network data to a published post.

Two servers, on purpose

The system runs on two separate cloud servers ("droplets"), and the split is deliberate.

SERVER	ROLE	WHY SEPARATE
`main droplet`	Holds the website network and all the source data (site descriptions, quality scores, the marketing-skills library). Builds the nightly content feed.	It is memory-constrained and already busy serving ~190 sites.
`hub droplet`	Runs the publishing brain: Postiz + its database, the scheduler, the approval portal, and the publisher that posts to social networks.	Publishing needs several always-on services (~5.6 GB); it gets its own 4 GB box so it never competes with the websites.

Plain English

One server grows and feeds the content; the other server publishes it. They talk over a secure connection once a night.

The nightly pipeline

Four scheduled jobs form a chain. Each hands off to the next. Times are UTC.

`05:00 · main droplet`

gen-social-feed.py reads the network data and writes a feed of publishable items, then copies it to the hub.

↓

`05:10 · hub`

generate.py turns feed items into scheduled posts using templates, rotating content so nothing repeats too soon.

↓

`every 10 min · hub`

publish_due.py takes any post whose time has arrived and sends it to the right networks — respecting the cost cap and the kill switch.

↓

`05:20 · hub`

report.py writes a digest of what published and pulls back basic performance numbers.

Every moving part

COMPONENT	WHAT IT IS	WHERE
Postiz	Open-source social publishing app — one place that can post to 30+ networks.	hub · port 5000
Temporal	The scheduler Postiz relies on to time posts. Runs as a lightweight dev server.	hub · port 7233
PostgreSQL	Postiz's database — stores connected accounts and their access tokens.	hub
Redis	Fast in-memory cache/queue Postiz uses internally.	hub
Portal (`/opt/portal`)	Our layer on top: the content queue, templates, generator, publisher, approval API, and reports.	hub
Feed generator	`gen-social-feed.py` — turns network data into publishable items.	main droplet
nginx + certbot	Web server + free SSL certificates, so everything is served securely over HTTPS.	both
ntfy	Push notifications — the system pings a phone channel when something fails.	external

The GitHub Repositories page links the open-source project behind each of these.

From raw data to a live post

Follow one post end to end:

1. A site in the network has the description *"Complete guide to home water filtration…"* in the network data file.
2. The feed generator packages it as a **site-feature** item with a verified description and URL.
3. The generator picks a matching template and produces the finished text, scheduled for an open time slot.
4. When that time arrives, the publisher signs the request and posts it to X and Bluesky, recording the live URLs.
5. The report job logs the post and its early engagement.

The guarantee

If a fact needed by a template is missing from the source data, the post is **held**, not guessed. Honesty is enforced by the machine, not by hoping the writer got it right.

02

The Content Engine

How real posts are written from real data — four content families, deterministic templates, and the honesty guard that never guesses.

Want to change the content?

This page explains how the engine writes posts. For the exact steps to edit wording, add a new post type, or refresh the feed, see Playbooks 6–8.

Four content families

Every post belongs to one of four families. Each is fed by a real, independent data source.

FAMILY	WHAT IT POSTS	FED BY
Site feature	Spotlights a site in the network with its real description and link.	the network site list (~177 sites)
AI-marketing tip	A genuine marketing principle drawn from our skills library, with a link to more.	the marketing-skills library (47 skills)
Milestone / win	Announces a real improvement in a site's quality score ("up 30 this week").	weekly quality-score snapshots
New-site spotlight	Introduces a site newly added to the network.	score snapshots (new entries)

Posting rotates across families so a feed never turns into all-promo or all-tips.

The honesty guard

This is the single most important design decision, and the thing that makes the system safe to run for clients. Every template declares the exact facts it needs. Before a post is written, the engine checks those facts are present in the source data.

- All facts present → the post is written.
- A fact missing → the post is **held** and flagged, never filled with a guess.

> **Why this matters to a client**
>
> A price, a square-footage, a "now selling" status — the system will only ever state what the source verifies. You can put this in front of a builder and it will not embarrass you by inventing a number.

Deterministic templates (and why they cost nothing)

Posts are produced by fixed templates that slot verified facts into proven sentence patterns — not by calling a paid AI model every time. That has two big consequences:

- **Zero per-post AI cost.** Generating 1,000 posts costs the same as generating one: nothing.
- **Predictable, on-brand output.** You always know what a post will read like before it goes out.

Each family has multiple template variants for variety. Editing or adding one is a small, safe code change.

WHERE THE TEMPLATES LIVE

```
/opt/portal/portal_templates.py       # builder-listing templates (demo)
/opt/portal/portal_templates_wt.py    # network self-promo templates
```

The feed generator

`gen-social-feed.py` on the main droplet is where real data becomes publishable items. It:

- reads the site list, decodes and cleans each description, drops anything truncated;
- pulls a genuine principle from each marketing skill (preferring the skill's stated goal);
- compares this week's quality scores against last week's and keeps only meaningful gains (+5 or more, score 60+);
- writes the combined feed and copies it to the hub.

It runs nightly at 05:00 UTC, just before the hub builds the day's posts. See Data Sources for exactly what it reads.

Rotation & variety

Two rules keep the feed fresh:

- **Least-recently-used first** — the content combination that hasn't been posted in the longest time is chosen next, so the system works through the whole network before repeating.
- **Family cap per batch** — no single family can dominate a batch, so tips, features, and wins stay mixed.

Adding or changing content

Common changes, in order of how often you'll make them:

YOU WANT TO…	WHERE
Reword posts / add a template variant	`portal_templates_wt.py`
Change what data becomes posts	`gen-social-feed.py` (main droplet)
Change how often a tenant posts	`clients.json` → `cadence`
Add a whole new content family	new template block + a feed section that emits its items

03

Connecting Social Platforms

Every network the hub can post to, what each one costs, what it needs to connect, and the exact steps to wire it up.

Just want the steps?

Go straight to Playbook 1 — Connect a social account for the exact click-by-click for each network. This page explains the landscape and what each connection needs.

Platform status & cost at a glance

`on = connected` `gate = needs your login / review`

PLATFORM	STATUS	WHAT CONNECTING NEEDS	COST TO POST
Bluesky	connected	An app password — no developer app.	Free
X / Twitter (@springnet)	connected	A developer app at console.x.com (OAuth).	Paid — ~1.5¢/post, ~20¢ with a link
Mastodon	app ready	Connect the account in the hub UI.	Free
LinkedIn	gated	Developer app + app review.	Free
Facebook + Instagram	gated	Meta developer app + business verification + app review.	Free
Threads, Pinterest, TikTok, YouTube, Reddit…	available	Each: a developer app; some need review.	Mostly free

The real gate is never "can it post?"

Posting is a solved problem. The one-time cost for each platform is creating the developer app and, for Meta/LinkedIn, passing their app review — a multi-day approval. Start those early.

How a connection actually works

Two patterns cover every platform:

1 · APP-PASSWORD PLATFORMS (EASIEST) — BLUESKY, MASTODON

You generate a password in the account settings and paste it in. No developer app, no review.

2 · OAUTH PLATFORMS — X, LINKEDIN, META, MOST OTHERS

You create a developer app on the platform, tell it our callback address, and paste the app's keys into the hub. Clicking "connect" then does a secure handshake that stores an access token.

The callback address is always the same shape:

```
https://postiz.wholetech.com/integrations/social/<platform>
```

e.g. `…/x`, `…/linkedin`, `…/facebook`. Whitelist it in the platform's app settings.

Where the keys go (and how they stay safe)

Platform keys never travel through chat. They are installed by short prompts you run yourself in your own terminal:

```
# Meta (Facebook + Instagram)
ssh -t root@64.227.29.192 /opt/postiz/set-meta-keys.sh

# LinkedIn
ssh -t root@64.227.29.192 /opt/postiz/set-linkedin-keys.sh
```

Each script prompts for the keys, writes them to a locked-down file (`/opt/postiz/social.env`, readable only by root), and restarts the hub. See Security for the full policy.

A word on X billing

X is the one paid channel

X closed its free posting tier in early 2026. Posting costs roughly 1.5¢ for text and ~20¢ when a post includes a link. The system caps X at a few posts per day so a small credit lasts months; every other connected network is free. See Cost control.

04

Daily Operations

Running the machine day to day: the schedule, the kill switch, cost control, cadence, logs, and manual posting.

Looking for the how-to?

This page is the day-to-day reference. For numbered, step-by-step procedures with the exact commands, see Playbooks.

The daily schedule

TIME (UTC)	WHERE	JOB	DOES
05:00	main	gen-social-feed.py	Build the day's content feed, ship to hub
05:10	hub	generate.py	Turn feed into scheduled posts
*/10 (every 10 min)	hub	publish_due.py	Publish anything due
05:20	hub	report.py	Write the digest
04:17	hub	backup.sh	Back up the hub

The kill switch

Freeze all posting instantly

Create one file and nothing publishes on any tenant until you remove it:

```
# stop everything
ssh root@64.227.29.192 touch /opt/portal/PAUSE

# resume
ssh root@64.227.29.192 rm /opt/portal/PAUSE
```

The publisher checks for this file first, every run. Use it any time something looks wrong — it is instant and total.

Cost control

The whole system costs **$24/month** to run (the hub server). Content generation is free. The only per-post cost is X.

CONTROL	SETTING	EFFECT
X daily cap	`X_DAILY_CAP` in `publish_due.py` (default 3)	Never more than N paid X posts/day; Bluesky is unaffected and free.
Cadence	`cadence` per tenant in `clients.json`	How many posts a tenant keeps in flight (~1/day at 7).
Channels	`channels` per tenant	Drop `"x"` to go 100% free.

Hard rule

The system runs on a flat subscription. It never uses per-token AI billing — content is template-generated, so audits and scale can't blow a budget.

Changing cadence or channels

```
# edit the tenant config on the hub
ssh root@64.227.29.192 nano /opt/portal/clients.json
# then regenerate (safe to run anytime)
ssh root@64.227.29.192 python3 /opt/portal/generate.py
```

Set `"require_approval": true` to route a tenant through the approval queue instead of auto-posting.

Posting on demand

To push something immediately, add it to the queue with its time set to now and run the publisher:

```
ssh root@64.227.29.192 python3 /opt/portal/publish_due.py
```

The publisher processes any post whose scheduled time has passed.

Monitoring & logs

LOG	WHAT
/var/log/social-feed.log	Nightly feed build (main droplet)
/var/log/portal-gen.log	Post generation
/var/log/portal-pub.log	Publishing results + live URLs
/var/log/portal-rep.log	Reports

Failures also push a phone notification via ntfy, so you hear about a problem without watching logs.

05

Playbooks — Every Task, Step by Step

The how-to heart of the manual: numbered, plain-English procedures for every operation — connecting accounts, approving posts, editing content, onboarding a client — each with the exact clicks and commands.

How to use this page

This is the true how-to guide. Every operation you can perform is a numbered playbook below — first in plain English (what you're doing and why), then the exact steps, then the commands for whoever runs the server. You do not need to be technical to follow the plain-English steps. Two machines are involved: the
hub
(does the posting) at `ssh root@64.227.29.192`, and the
main server
(holds the website data) at `ssh root@143.198.182.180`. A line starting with `ssh` is typed into a terminal.

#	PLAYBOOK	WHEN YOU NEED IT
—	The lifecycle of one post	To understand the whole flow first
1	Connect a social account	Adding X, Bluesky, Mastodon, LinkedIn, or Meta
2	Approve, edit, or skip a post	Running an approval-gated account
3	Post something right now	You want an immediate post
4	Change how often it posts	More or fewer posts
5	Add or remove a network	Turning a channel on/off for a tenant
6	Change what a post says	Editing the wording/templates
7	Add a new type of post	A new content family
8	Refresh the content now	Pull fresh data immediately
9	Preview what's queued	See posts before they go out
10	Stop everything instantly	Something looks wrong
11	See what went out	Checking results

The lifecycle of one post — and where you step in

Before the playbooks, here is the whole journey of a single post. Five stages run automatically on a schedule; the two **green** markers are where a human can step in.

`Stage 1 · 05:00 · gather`

The main server builds the feed. It reads your real data (site descriptions, marketing skills, score changes) and writes a clean list of publishable items. **You can step in:** edit the data sources or the feed builder (Playbook 8, Playbook 6).

↓

`Stage 2 · 05:10 · draft`

The hub drafts the posts. It runs each feed item through a template to produce finished wording, and schedules each into an open time slot. If a required fact is missing, the post is **held**, never guessed.

↓

`Stage 3 · the gate`

Approve — or auto-approve. If the tenant is set to require approval, drafts wait in the approval queue. **You step in:** approve, edit, or skip (Playbook 2). If the tenant is set to auto-fire, drafts skip this gate and are approved automatically.

↓

`Stage 4 · every 10 min · publish`

The hub publishes what's due. Any approved post whose time has arrived is sent to its networks — within the X spend cap, and only if the kill switch is off.

↓

`Stage 5 · 05:20 · report`

The hub logs results. What published, where, and early engagement — written to the logs and the digest.

The one thing to remember

Nothing you do can make the system state an unverified fact. The worst case is a post is *held* and doesn't go out — never that it invents something. That is what makes it safe to run on autopilot and safe to run for a client.

Playbook 1 — Connect a social account

Every network is added the same way in spirit: authorize the account once, and the hub can post to it forever. Two kinds of networks: **app-password** ones (easiest) and **developer-app** ones (a one-time setup, sometimes an approval wait). Do the easy ones first.

Bluesky (easiest — 3 minutes, free)

> **In plain English**
> Bluesky lets you create a one-time "app password" that you paste into the hub. No developer account, no approval.

DO IT — STEP BY STEP

1. In a browser, sign in to **bsky.app** with the account you want to post from.
2. Go to **Settings** → **Privacy and security** → **App passwords**, click **Add app password**, name it "WholeReach", and copy the password it shows (you only see it once).
3. Open the hub at postiz.wholetech.com and sign in.
4. Click **Add channel** → **Bluesky**. Enter service `https://bsky.social`, your handle (e.g. `walhus.bsky.social`), and paste the app password. Click connect.
5. Done — a test post confirms it. The channel now shows in the hub.

UNDER THE HOOD (FOR THE TECHNICAL READER)

The hub stores the returned access token in the `Integration` table; the publisher reads it directly. Nothing to install.

Mastodon (easy — free)

> **In plain English**
> The hub's Mastodon app is already registered. You just authorize your account through your Mastodon server.

DO IT — STEP BY STEP

1. Sign in to the hub at postiz.wholetech.com.
2. Click **Add channel** → **Mastodon**. Enter your instance URL (e.g. `https://mastodon.social`).
3. You're bounced to Mastodon to log in and click **Authorize**. You return to the hub and the channel is connected.

X / Twitter (developer app — paid per post)

In plain English

X needs a one-time developer app so the hub can post on your behalf. X also charges per post now, so you load a small credit. Everything after the first setup is automatic.

DO IT — STEP BY STEP

1. Go to **console.x.com**, sign in with the account you'll post from, and create a project + app. Set the app to **Read and Write** and type "Web App / Automated App or Bot".
2. In the app's **User authentication settings**, set the callback URL to `https://postiz.wholetech.com/integrations/social/x`.
3. Copy the app's **API Key** and **API Secret**.
4. Install the keys on the hub — run this in your own terminal and paste the keys when prompted (they never pass through anyone else):

```
ssh -t root@64.227.29.192 /opt/postiz/set-x-keys.sh
```

5. Back in the hub UI, click **Add channel** → **X** and authorize.
6. Load credit: in the X developer portal, add funds (a $25 credit lasts months at our cap). See Playbook 11 for the spend cap.

UNDER THE HOOD (FOR THE TECHNICAL READER)

Auth is OAuth 1.0a; keys live in `/opt/postiz/social.env` (root-only). The publisher signs each request with the stored token.

LinkedIn (developer app + review — free)

In plain English

Same idea as X, but LinkedIn reviews your app first (a few days). Start it early.

DO IT — STEP BY STEP

1. Go to **developer.linkedin.com**, create an app, and link it to a LinkedIn Page you control.
2. Request the **Community Management API** (page posting). This triggers LinkedIn's review.
3. Set the callback to `https://postiz.wholetech.com/integrations/social/linkedin`.
4. When approved, copy the Client ID + Secret and install them:

```
ssh -t root@64.227.29.192 /opt/postiz/set-linkedin-keys.sh
```

5. In the hub, **Add channel** → **LinkedIn** and authorize.

Facebook + Instagram (Meta app + business review — free)

In plain English

Meta is the most involved: one app covers both Facebook and Instagram, but it needs business verification and app review. Begin this first because the wait is longest.

DO IT — STEP BY STEP

1. Go to **developers.facebook.com**, create an app (type “Business”), and add the **Instagram Graph API** and **Pages** products.
2. Complete **Business Verification** and request the permissions to post to a Page and its linked Instagram professional account. Provide the privacy policy (automarketingengine.com/privacy/) and the data-deletion URL (the `#data-deletion` section on that page).
3. Set the callback to `https://postiz.wholetech.com/integrations/social/facebook`.
4. After approval, copy the App ID + Secret and install them:

   ```
   ssh -t root@64.227.29.192 /opt/postiz/set-meta-keys.sh
   ```

5. In the hub, **Add channel** → **Facebook**, authorize, and pick the Page (and its linked Instagram account).

UNDER THE HOOD (FOR THE TECHNICAL READER)

Plan the wait

Meta and LinkedIn reviews can take several business days. Connecting Bluesky, Mastodon, and X gets you live immediately while those are in review.

Playbook 2 — Approve, edit, or skip a post

Approve, edit, or skip

> **In plain English**
>
> On accounts set to require approval, drafts wait in a private queue. You (or the client) open a secure link on a phone and tap Approve, Edit, or Skip. Only approved posts ever publish.

DO IT — STEP BY STEP

1. Open the approval link for the account (how to create one: Managing Clients). No password — the link itself is the key.
2. You see a stack of cards, one per drafted post: the text, the network(s), and the scheduled time.
3. For each card, tap **Approve** (it will publish at its time), **Edit** (change the wording, then approve), or **Skip** (it won't publish; you can note why).
4. Held cards (missing a verified fact) are shown but can't be approved until the fact is supplied.
5. That's it — approved posts publish automatically at their scheduled time.

UNDER THE HOOD (FOR THE TECHNICAL READER)

The queue is served by `portal_api.py` (`POST /portal-api/act` with `approve|edit|skip`). Operators can also flip a post directly:

```
ssh root@64.227.29.192 "python3 -c \"import
sqlite3;c=sqlite3.connect('/opt/portal/portal.db');c.execute('UPDATE posts SET state=1 WHERE
id=123'.replace('1',chr(39)+'approved'+chr(39)));c.commit()\""
```

(Simpler: use the link. The publisher only ever sends posts in state `approved`.)

Playbook 3 — Post something right now

Publish immediately

> **In plain English**
>
> Normally posts go out at their scheduled slot. To push one out now, move its time to “now” and run the publisher.

DO IT — STEP BY STEP

1. Find the post you want (see Playbook 9 to list the queue and get its ID).
2. Set its time to now and make sure it's approved, then run the publisher:

   ```
   ssh root@64.227.29.192 "python3 - <<PY
   import sqlite3
   c=sqlite3.connect('/opt/portal/portal.db')
   c.execute("UPDATE posts SET state='approved', scheduled=strftime('%Y-%m-%dT%H:%M:00Z','now','-1
   minute') WHERE id=123")
   c.commit()
   PY"
   ssh root@64.227.29.192 python3 /opt/portal/publish_due.py
   ```

   Replace `123` with your post's ID.
3. The publisher prints the live URL(s) it created. Done.

UNDER THE HOOD (FOR THE TECHNICAL READER)

`publish_due.py` selects `state='approved' AND scheduled ⩽ now`, so setting the time in the past makes it eligible on the next run. The X daily cap and the `PAUSE` kill switch still apply.

Playbook 4 — Change how often it posts

Turn the posting rate up or down

> **In plain English**
>
> Each account has a "cadence" — how many posts it keeps in flight. Higher = more posts. Change one number, then regenerate.

DO IT — STEP BY STEP

1. Open the tenant config on the hub:

   ```
   ssh root@64.227.29.192 nano /opt/portal/clients.json
   ```

2. Find the account and change `"cadence"` (7 ≈ one a day; 14 ≈ two a day). Save (`Ctrl+O`, `Enter`) and exit (`Ctrl+X`).
3. Rebuild the queue (safe to run anytime):

   ```
   ssh root@64.227.29.192 python3 /opt/portal/generate.py
   ```

4. Check it with Playbook 9.

UNDER THE HOOD (FOR THE TECHNICAL READER)

`generate.py` tops the queue up to `cadence` open posts, spread across the week's time slots, with least-recently-used rotation and a per-family cap for variety.

Playbook 5 — Add or remove a network for an account

Turn a channel on or off

> **In plain English**
>
> Which networks an account posts to is a simple list. Add a name to post there; remove it to stop. Bluesky is free; X costs per post.

DO IT — STEP BY STEP

1. Make sure the network is connected first (Playbook 1).
2. Edit the tenant config:

```
ssh root@64.227.29.192 nano /opt/portal/clients.json
```

3. Set `"channels"` to the networks you want, e.g. `["x","bluesky"]`. Drop `"x"` to go 100% free. Use `["dry"]` to draft without sending anywhere (for testing). Save and exit.
4. Regenerate so new drafts carry the new channels:

```
ssh root@64.227.29.192 python3 /opt/portal/generate.py
```

UNDER THE HOOD (FOR THE TECHNICAL READER)

> **Never point a test tenant at real channels.**
>
> Use `["dry"]` for anything experimental — it logs the post without sending it.

Playbook 6 — Change what a post says

Edit the wording (templates)

> **In plain English**
>
> Posts are written from a handful of sentence patterns called templates. Editing one changes how all posts of that type read. This is a small, safe text change.

DO IT — STEP BY STEP

1. Open the templates file on the hub:

   ```
   ssh root@64.227.29.192 nano /opt/portal/portal_templates_wt.py
   ```

2. Each template is one line in the `TEMPLATES` list, shaped like:

   ```
   (kind, family, id, required fields, how to write it)
   ("site", "site_feature", "sf1", ["blurb","url"],
    lambda l, b: _cap("%s %s" % (_trim(l["blurb"],210), l["url"])))
   ```

   Change the wording inside the final part. Keep the `%s` placeholders and the fields listed in "required fields".
3. Save and exit. Rebuild:

   ```
   ssh root@64.227.29.192 python3 /opt/portal/generate.py
   ```

4. Preview the result (Playbook 9) before it goes out.

UNDER THE HOOD (FOR THE TECHNICAL READER)

Templates are deterministic (no AI call), so output is predictable and costs nothing to generate. A field can only be used if it's in the item's data — that's the honesty guard.

Playbook 7 — Add a whole new type of post

Create a new content family

> **In plain English**
>
> A “type” of post (site feature, tip, milestone…) is a family. Adding one means telling the system both how to write it and where its facts come from.

DO IT — STEP BY STEP

1. **Add the writing pattern.** In `portal_templates_wt.py`, add one or more template lines with a new `kind` and `family`, listing the fields they need.
2. **Supply the facts.** In the feed builder on the main server (`/root/gen-social-feed.py`), add a section that reads your data and emits items tagged with that same `kind` and those fields.
3. Rebuild the feed and the queue:

```
ssh root@143.198.182.180 python3 /root/gen-social-feed.py
ssh root@64.227.29.192 python3 /opt/portal/generate.py
```

4. Preview (Playbook 9).

UNDER THE HOOD (FOR THE TECHNICAL READER)

Keep the rule: emit an item only when every field its template needs is present. Missing field → the post is held, never guessed.

Playbook 8 — Refresh the content right now

Pull fresh data immediately

> **In plain English**
>
> The feed refreshes itself nightly. To pull the latest data right now (new sites, new score wins), run the two builders by hand.

DO IT — STEP BY STEP

1. Rebuild the feed on the main server (this also ships it to the hub):

```
ssh root@143.198.182.180 python3 /root/gen-social-feed.py
```

2. Rebuild the queue on the hub:

```
ssh root@64.227.29.192 python3 /opt/portal/generate.py
```

3. Preview (Playbook 9).

Playbook 9 — See what's queued before it publishes

List the upcoming posts

> **In plain English**
>
> You can look at every drafted/approved post that hasn't gone out yet — the exact text, its network, and when it will publish.

DO IT — STEP BY STEP

1. Run this on the hub to print the pending/approved queue:

```
ssh root@64.227.29.192 "python3 - <<PY
import sqlite3
c=sqlite3.connect('/opt/portal/portal.db'); c.row_factory=sqlite3.Row
for r in c.execute("SELECT id,client,state,substr(scheduled,1,16) s,text FROM posts WHERE state
IN ('pending','approved') AND scheduled>=datetime('now') ORDER BY scheduled"):
    print('#%d [%s/%s] %s'%(r['id'],r['client'],r['state'],r['s'])); print('   ',r['text'][:130])
PY"
```

2. Each line shows the post ID (use it in other playbooks), the account, whether it's pending or approved, its time, and the text.

Playbook 10 — Stop everything instantly

The kill switch

> **In plain English**
>
> One command freezes all publishing on every account until you turn it back on. Use it any time something looks wrong — it's instant and total.

DO IT — STEP BY STEP

1. Freeze:

```
ssh root@64.227.29.192 touch /opt/portal/PAUSE
```

2. Nothing will publish while that file exists (the publisher checks for it first, every run).
3. Resume when ready:

```
ssh root@64.227.29.192 rm /opt/portal/PAUSE
```

Playbook 11 — See what went out (and manage X spend)

Check results

In plain English

Every publish is logged with the live URL. You can read the log or the digest.

DO IT — STEP BY STEP

1. See recent publishes and their URLs:

```
ssh root@64.227.29.192 tail -n 40 /var/log/portal-pub.log
```

2. See the nightly feed build:

```
ssh root@143.198.182.180 tail /var/log/social-feed.log
```

3. Failures also push a phone alert via ntfy, so you hear about problems without watching logs.

Manage X spend

In plain English

X is the only paid network (~1.5¢ a post, ~20¢ with a link). The system caps X at 3 posts a day so a small credit lasts months; Bluesky and the rest are free and uncapped.

DO IT — STEP BY STEP

1. To change the cap, edit `X_DAILY_CAP` near the top of `/opt/portal/publish_due.py` on the hub.
2. To post more to X, raise the cap and add funds in the X developer portal.
3. To stop paying X entirely, remove `"x"` from a tenant's channels (Playbook 5) — Bluesky keeps posting free.

06

Managing Clients — Selling It as a Service

How the same engine manages other people's social media: multi-tenant setup, onboarding, the approval portal, pricing, and white-label.

Yes — this manages other people's social media today.

The engine is multi-tenant by design. Each client is one entry in a config file, with their own connected accounts, cadence, content, and (optionally) an approval step. Our own network is simply the first tenant.

The multi-tenant model

Every client — including us — is a "tenant". A tenant is defined by a single block in `/opt/portal/clients.json`:

```
"acme-builders": {
  "name": "Acme Builders",
  "brand": { "brand_name": "Acme", "tagline": "Built for life." },
  "cadence": 5,                    # posts kept in flight
  "channels": ["bluesky","linkedin"],  # their connected networks
  "require_approval": true,        # review before posting
  "template_module": "portal_templates_wt",
  "listings": "/opt/portal/listings/acme-builders.json"
}
```

Golden rule

Never point a demo/test tenant at a real client's channels. Test tenants use the `"dry"` channel, which logs a post without sending it anywhere.

Client onboarding — step by step

Add a new client, start to finish

> **In plain English**
>
> A client is one entry in a config file plus a file of their verified facts. This walkthrough takes a fictional client "Acme Builders" from nothing to live, safely (a dry run first, then real posting).

DO IT — STEP BY STEP

1. **Discovery.** Agree what they want posted, how often, on which networks, and whether they want to approve each post or run hands-off.
2. **Connect their accounts.** Have them authorize their networks through the hub — follow Playbook 1 once per network. Their tokens are stored under their own account.
3. **Create their facts file** on the hub — the verified things their posts can draw from (products, listings, services, wins). Model it on an existing one:

   ```
   ssh root@64.227.29.192 nano /opt/portal/listings/acme-builders.json
   ```

4. **Add their tenant block** to the config:

   ```
   ssh root@64.227.29.192 nano /opt/portal/clients.json
   ```

   Add an entry like:

   ```
   "acme-builders": {
     "name": "Acme Builders",
     "brand": { "brand_name": "Acme", "tagline": "Built for life." },
     "cadence": 5,
     "channels": ["dry"],            # dry first — see step 5
     "require_approval": true,
     "template_module": "portal_templates_wt",
     "listings": "/opt/portal/listings/acme-builders.json"
   }
   ```

5. **Dry run.** With channels set to `["dry"]`, generate and review the drafts — nothing is sent anywhere:

   ```
   ssh root@64.227.29.192 python3 /opt/portal/generate.py
   ```

   Then read them with Playbook 9.
6. **Go live.** When the drafts look right, change `"channels"` to their real networks (e.g. `["bluesky","linkedin"]`) and regenerate. With `require_approval: true`, posts now wait in

their queue.

7. **Give them their approval link** (next section) — or approve on their behalf.

UNDER THE HOOD (FOR THE TECHNICAL READER)

> **Always dry-run first.**
> The `["dry"]` channel logs a post without sending it — the safe way to prove a new client's output before anything reaches their real audience.

The approval portal

For clients who want a say, each tenant has a private, phone-friendly approval queue. The client opens a secure magic link — no password — and taps **Approve**, **Edit**, or **Skip** on each drafted post. Only approved posts publish.

- Links are signed and time-limited (they expire).
- Skips can capture a reason, which feeds back into better drafting.
- Held posts (missing a verified fact) show up clearly and cannot be published until resolved.

This is the screen that sells the service: the client sees a tidy stream of on-brand posts and approves a week in two minutes.

Create a client's approval link

> **In plain English**
> Each client gets their own private, expiring link — no login. You generate it once and send it to them.

DO IT — STEP BY STEP

1. Generate a 30-day link for the client (use their tenant name):

```
ssh root@64.227.29.192 "python3 -c \"import sys;sys.path.insert(0,'/opt/portal');import portal_db as db;print('https://postiz.wholetech.com/portal/app/?c=acme-builders&t='+db.make_token('acme-builders',30))\""
```

2. It prints the full link. Send it to the client (or open it yourself to approve for them).
3. When it expires, run the command again for a fresh one.

UNDER THE HOOD (FOR THE TECHNICAL READER)

Tokens are HMAC-signed with the server secret and carry an expiry — they can't be forged or reused after they lapse.

Brand safety for client accounts

- **The honesty guard** (see Content Engine) means the system never states an unverified fact about a client's business.
- **Approval mode** puts a human in the loop for sensitive accounts.
- **The kill switch** and per-tenant channels let you stop or narrow any client instantly.
- **Account isolation** — each tenant's tokens and content are separate; one client can never post to another's accounts.

Pricing the service

Because generation is free and the only variable cost is X (pennies), margins are strong. Indicative managed-service tiers:

TIER	FOR	INDICATIVE PRICE*	INCLUDES
Starter	Solo / small business	$300–500/mo	1–2 networks, ~1 post/day, monthly report
Managed	Growing brands / builders	$500–1,500/mo	3–4 networks, approval queue, weekly report, content tuning
Specialist	Technical / niche verticals	$1,000–2,500/mo	Custom content families, white-label, deeper reporting

*Estimates for planning, not a rate card. Set final pricing per engagement.

White-label

The approval portal and reports can be served under a partner's branding. Client-facing pages carry the agency's name; the WholeTech engine stays behind the scenes. This is how a partner (for example, an agency serving home builders) resells the system as their own product.

07

Security, Privacy & Compliance

Secrets handling, access control, data ownership, platform policy, disclosure law, backups, and disaster recovery.

Secrets handling

- Platform keys live in `/opt/postiz/social.env`, permission `600` (root-only), loaded into the app as environment variables.
- Keys are installed by self-run scripts (`set-*-keys.sh`) — they never pass through chat, email, or an assistant.
- Access tokens for connected accounts are stored in the hub database, not in code.
- If a key is ever exposed, rotate it at the platform and re-run the installer.

Access control

- **Servers** are reachable only by SSH key — no passwords.
- **Hub sign-up is disabled** (`DISABLE_REGISTRATION=true`) now that the owner account exists, so no one can self-register.
- **Client approval links** are HMAC-signed and expire — they can't be forged or reused indefinitely.
- **HTTPS everywhere** via Let's Encrypt (certbot); the login cookie only works over HTTPS.

Data ownership & privacy

- Each client's content and tokens are isolated to their tenant.
- Public privacy & terms pages are published (required for Meta review), including a data-deletion path.
- We store only what's needed to publish and report; we don't harvest audience data.
- A client can be fully removed by deleting their tenant block, listings file, and connected accounts.

Compliance & platform policy

AREA	WHAT TO KNOW
FTC disclosure	Paid or affiliate promotion must be disclosed. Keep endorsements truthful — the honesty guard helps, but disclosure is an editorial responsibility.
Platform Terms	Automated posting is allowed within each platform's API rules; avoid spammy volume. Our cadence is deliberately modest.
Email/CAN-SPAM	If you add newsletters, include a real unsubscribe and sender address.
Data protection	Handle any client/customer data on a need-to-use basis; honor deletion requests.

See the Resource Links page for the primary-source policy documents.

Backup & disaster recovery

- **Nightly backups** run on the hub (`backup.sh`, 04:17 UTC), covering the database, the portal data, and config.
- Backups are kept on a schedule and copied off-box (cloud object storage).
- **To rebuild the hub** from scratch: provision a new server, restore the Docker stack and the portal directory, restore the database, re-point DNS. The connected-account tokens come back with the database restore.
- **The websites** and source data on the main droplet have their own separate nightly backup.

> **Lesson learned**
>
> Keep the servers on manual OS updates so an unattended reboot never interrupts a run. Long jobs should survive a restart, not depend on the box never rebooting.

Incident response

1. **Contain** — hit the kill switch.
2. **Diagnose** — check the logs and the ntfy alerts (see Troubleshooting).
3. **Rotate** — if credentials are involved, rotate keys and re-install.
4. **Resume** — remove the PAUSE file once resolved.

08

Data Sources

Every source of truth the content engine reads from — where it lives, what it provides, and how often it refreshes.

Every source the engine reads

SOURCE	PROVIDES	LOCATION	REFRESH
Network site list	Each site's domain + verified description → site features & spotlights	`main:/root/wt-sites.json`	As the network changes
Marketing-skills library	47 skills → AI-marketing tips	`skills-data.js`	Weekly
Quality-score snapshots	Daily AI-readiness scores → milestones & new-site spotlights	`main:/root/ama-snapshots/`	Daily
The built feed	The combined, cleaned, publishable items	`hub:/opt/portal/listings/wholetech.json`	Nightly
Connected-account tokens	Credentials to publish to each network	`hub database`	On connect/refresh

One principle governs all sources

Verified in, or held out.

A field only becomes part of a post if it is present in the source. Truncated descriptions are dropped; missing scores are skipped; unverifiable facts are held. The feed is deliberately conservative — it would rather post less than post something it can't stand behind.

Adding a new source

To feed a new kind of post, add a section to `gen-social-feed.py` that reads the new data, cleans it, and emits items tagged with a new `kind`; then add matching templates. Keep the same rule: emit an item only when every field a template needs is verified.

09

GitHub Repositories

The open-source building blocks behind every aspect of the system, grouped by function, with links.

The open-source projects behind every layer of the system, grouped by what they do. Everything here is self-hostable — no proprietary lock-in.

Social publishing / scheduling hubs

PROJECT	LANGUAGE	WHAT IT DOES
gitroomhq/postiz-app	TypeScript	Self-hosted social scheduler for 30+ platforms with AI + analytics (Buffer alternative). This is our publishing hub.
inovector/mixpost	PHP	Laravel package to schedule, publish & manage social content on your own server. Our documented fallback.
ayrshare/social-media-api	JavaScript	Node SDK for the Ayrshare posting/analytics API (the platform itself is commercial SaaS; only the SDK is OSS).

Workflow / scheduling engines

PROJECT	LANGUAGE	WHAT IT DOES
temporalio/temporal	Go	Durable execution engine for reliable, retryable long-running workflows. Powers post scheduling in our hub.
n8n-io/n8n	TypeScript	Fair-code workflow automation with 400+ integrations & AI nodes. Popular visual glue for posting pipelines.
node-red/node-red	JavaScript	Flow-based low-code wiring of APIs and services.
windmill-labs/windmill	Rust/TypeScript	Turn scripts into workflows, schedules & internal UIs.

Bluesky / AT Protocol

PROJECT	LANGUAGE	WHAT IT DOES
bluesky-social/atproto	TypeScript	Official AT Protocol packages/SDK (@atproto/api). The basis of our Bluesky posting.
bluesky-social/social-app	TypeScript	Reference Bluesky client app (web + React Native) — good API usage patterns.

Mastodon

PROJECT	LANGUAGE	WHAT IT DOES
halcy/Mastodon.py	Python	Python wrapper for the full Mastodon API.
neet/masto.js	TypeScript	Universal Mastodon API client (REST + streaming).

Newsletter / email (self-hosted)

PROJECT	LANGUAGE	WHAT IT DOES
knadh/listmonk	Go	High-performance self-hosted newsletter & mailing-list manager.
mautic/mautic	PHP	Open-source marketing automation — campaigns, drip, lead scoring.
pentacent/keila	Elixir	Self-hosted newsletter tool (Mailchimp alternative).

Analytics (self-hosted)

PROJECT	LANGUAGE	WHAT IT DOES
umami-software/umami	TypeScript	Privacy-focused, lightweight web analytics.
plausible/analytics	Elixir	Simple, privacy-first Google Analytics alternative.
matomo-org/matomo	PHP	Full-featured GA alternative with goals, funnels, attribution.
PostHog/posthog	Python/TS	Product analytics, funnels, session replay, feature flags.

Link shortening / tracking

PROJECT	LANGUAGE	WHAT IT DOES
dubinc/dub	TypeScript	Modern link-management platform with click analytics & UTM builder.
thedevs-network/kutt	TypeScript	Self-hosted URL shortener with stats and API.
shlinkio/shlink	PHP	Self-hosted URL shortener with rich visit analytics & REST API.

Uptime / monitoring

PROJECT	LANGUAGE	WHAT IT DOES
louislam/uptime-kuma	JavaScript	Self-hosted uptime monitor with status pages & alerts.
TwiN/gatus	Go	Developer-oriented health dashboard, config-as-code.

Notification

PROJECT	LANGUAGE	WHAT IT DOES
binwiederhier/ntfy	Go	Simple pub/sub push notifications via HTTP. We use it for failure alerts.
caronc/apprise	Python	One library to push notifications to 100+ services.

Browser automation (connect flows)

PROJECT	LANGUAGE	WHAT IT DOES
microsoft/playwright	TypeScript	Cross-browser automation & testing — automates OAuth/connect flows.
puppeteer/puppeteer	TypeScript	Headless/headful Chrome control API.
browser-use/browser-use	Python	Let LLM agents drive a real browser.

Content generation / LLM orchestration

PROJECT	LANGUAGE	WHAT IT DOES
langchain-ai/langchain	Python	Framework for building LLM apps & chains.
langgenius/dify	TypeScript/Python	Self-hosted LLM app platform (agents, RAG, workflows).
FlowiseAI/Flowise	TypeScript	Drag-and-drop builder for LLM flows & agents.
run-llama/llama_index	Python	Data framework for RAG / context-augmented LLM apps.
crewAIInc/crewAI	Python	Framework for orchestrating role-playing multi-agent crews.
ollama/ollama	Go	Run open LLMs locally with a simple API — no per-token API cost.

SEO / sitemap tooling

PROJECT	LANGUAGE	WHAT IT DOES
towfiqi/serpbear	TypeScript	Self-hosted Google keyword rank tracker with GSC integration & JSON API.
ekalinin/sitemap.js	TypeScript	Streaming sitemap-generating framework + CLI for Node.

10

Video Library

Curated, embedded videos on marketing automation, AI marketing, and the platforms this system builds on.

The curated video library

Hand-picked, embed-verified explainers on marketing automation, AI marketing, and the platforms this system builds on.

VIDEO	CHANNEL	ABOUT
The Complete Guide to Marketing Automation with HubSpot	HubSpot Academy	Official overview of what marketing automation is and how to build it.
HubSpot Marketing Hub Explained: Forms, Automation, Campaigns & Reporting	The Gist — HubSpot Strategists	Current walkthrough of the automation, campaign and reporting toolset.
HubSpot Automation Tutorial: Workflows & Sequences Explained	Thalita Milan	Beginner-friendly breakdown of automated workflows vs. sequences.
HubSpot's AI Tools: Automate Your Marketing in Minutes	Evenbound	How AI features inside a marketing platform speed up routine work.
Proven Marketing Strategies To Create Explosive Growth	Leveling Up with Eric Siu	Established marketer on AI-era growth and modern strategy.
Automate Social Media Posts with n8n (Step-by-Step)	Hostinger Academy	Clean step-by-step build of a social-posting automation in n8n.
n8n Social Media Automation: AI Agent posts to Facebook, Instagram, LinkedIn, X	syncbricks	Building a self-hosted AI agent that posts across networks.
This n8n Automation will AUTOMATE your Social Media	Let's Automate It	End-to-end n8n social pipeline with a reusable template.
Self-Hosted Postiz Social Scheduler (Postiz + n8n + Docker)	Automation Hub Official	Self-hosting the open-source Postiz scheduler — the app our hub runs.
Postiz: Your New Self-Hosted Social Media Planner — No Fees!	DB Tech	Popular homelab channel installing and touring Postiz.
Postiz: Free Open Source Social Media Scheduler	Elestio	Concise intro to Postiz as a Buffer alternative you own.
How to Use Bluesky! (Beginner to Pro)	Matt Keil	Practical guide to Bluesky, the AT-Protocol decentralized network.
How to Use Mastodon: the COMPLETE GUIDE	The Linux Experiment	Thorough guide to Mastodon and the fediverse for newcomers.
Content Repurposing: Turn One Video Into 10 Pieces of Content	Patricia Kelikani	The content-engine idea: one asset repurposed across formats at scale.
How To Measure Social Media ROI	HubSpot Marketing	Official explainer on the social ROI formula and tracking.

VIDEO	CHANNEL	ABOUT
Use Marketing Campaigns to Track ROI and Prove What's Working	The Gist — HubSpot Strategists	Hands-on campaign-level ROI/analytics attribution walkthrough.

Watch them — embedded

Every video from the library above as a tap-to-play poster. (In the live web manual at wholereach.com/usermanual/videos.html these play inline.)

The Complete Guide to Marketing Automation with HubSpot

HubSpot Academy

Official overview of what marketing automation is and how to build it.

HubSpot Marketing Hub Explained: Forms, Automation, Campaigns & Reporting

The Gist — HubSpot Strategists

Current walkthrough of the automation, campaign and reporting toolset.

HubSpot Automation Tutorial: Workflows & Sequences Explained

Thalita Milan

Beginner-friendly breakdown of automated workflows vs. sequences.

HubSpot's AI Tools: Automate Your Marketing in Minutes

Evenbound

How AI features inside a marketing platform speed up routine work.

Proven Marketing Strategies To Create Explosive Growth

Leveling Up with Eric Siu

Established marketer on AI-era growth and modern strategy.

Automate Social Media Posts with n8n (Step-by-Step)

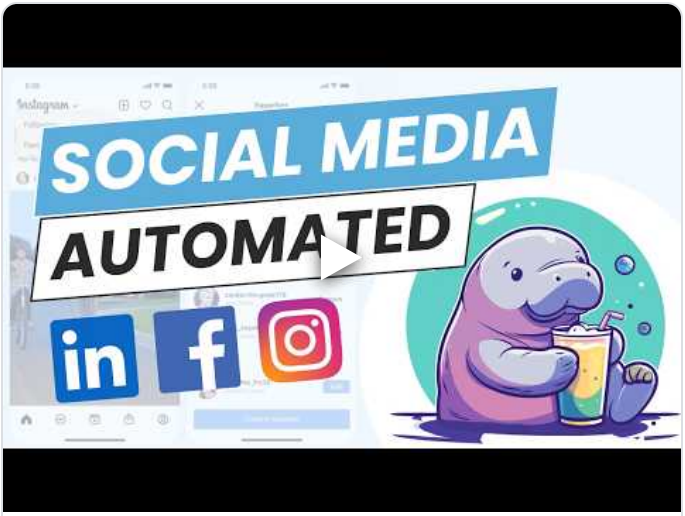
Hostinger Academy

Clean step-by-step build of a social-posting automation in n8n.

n8n Social Media Automation: AI Agent posts to Facebook, Instagram, LinkedIn, X

syncbricks

Building a self-hosted AI agent that posts across networks.



This n8n Automation will AUTOMATE your Social Media

Let's Automate It

End-to-end n8n social pipeline with a reusable template.

Self-Hosted Postiz Social Scheduler (Postiz + n8n + Docker)

Automation Hub Official

Self-hosting the open-source Postiz scheduler — the app our hub runs.

Postiz: Your New Self-Hosted Social Media Planner — No Fees!

DB Tech

Popular homelab channel installing and touring Postiz.

Postiz: Free Open Source Social Media Scheduler

Elestio

Concise intro to Postiz as a Buffer alternative you own.

How to Use Bluesky! (Beginner to Pro)

Matt Keil

Practical guide to Bluesky, the AT-Protocol decentralized network.

How to Use Mastodon: the COMPLETE GUIDE

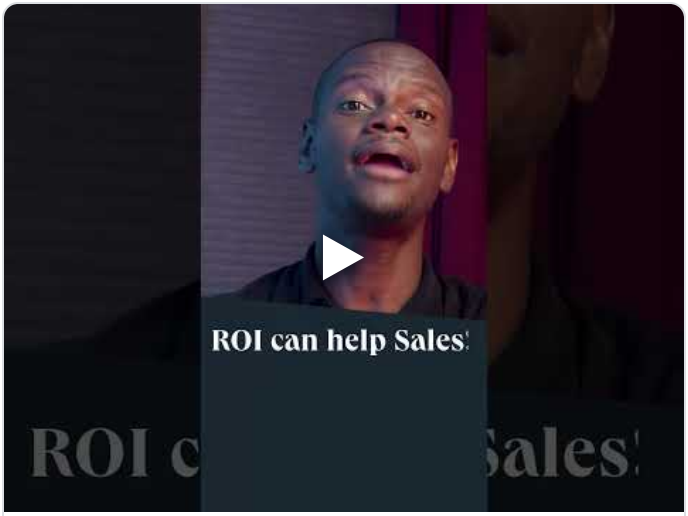
The Linux Experiment

Thorough guide to Mastodon and the fediverse for newcomers.

Content Repurposing: Turn One Video Into 10 Pieces of Content

Patricia Kelikani

The content-engine idea: one asset repurposed across formats at scale.



How To Measure Social Media ROI

HubSpot Marketing

Official explainer on the social ROI formula and tracking.

Use Marketing Campaigns to Track ROI and Prove What's Working

The Gist — HubSpot Strategists

Hands-on campaign-level ROI/analytics attribution walkthrough.

Resource Links

The best external references on automated and AI-driven marketing — docs, guides, and authorities worth bookmarking.

Authoritative external references worth bookmarking — grouped by topic.

Foundational guides & blogs

- **HubSpot Marketing Blog** — The most-referenced practitioner blog for inbound, automation, and campaign playbooks.
- **Buffer Resources** — Data-backed guides on social scheduling, cadence, and small-team automation.
- **Hootsuite Blog** — Platform-agnostic social strategy, algorithm changes, and posting tactics.
- **Sprout Social Insights** — Research-driven social marketing and analytics articles.
- **Social Media Examiner** — Long-running, tactical how-to library covering every major social channel.
- **Neil Patel Blog** — High-volume SEO/content/growth tutorials for solo and SMB marketers.
- **Marketing AI Institute** — The leading dedicated source on applying AI to marketing.

Platform developer docs

- **X (Twitter) Developer Platform** — Official X API v2 docs for posting, reading, and analyzing content.
- **Meta Graph API** — Core API for Facebook + Instagram publishing, insights, and page management.
- **Instagram Platform (Meta)** — Instagram-specific content publishing and messaging APIs.
- **LinkedIn Marketing API** — Official docs for LinkedIn page posting, campaigns, and ad reporting.
- **Bluesky / AT Protocol Docs** — Official developer docs for bots, feeds, and clients on Bluesky.
- **Mastodon API** — Full REST API reference for posting and reading on the fediverse.
- **Pinterest Developers** — Official API for creating pins, boards, and reading analytics.
- **TikTok for Developers** — Content Posting API, Login Kit, and embeds for TikTok automation.
- **YouTube Data API v3** — Programmatic upload, metadata, and analytics for YouTube.

AI for marketing

- **Anthropic — Prompt Engineering Overview** — Authoritative prompting techniques applicable to marketing copy generation.
- **Anthropic — Prompt Engineering for Business** — Business-focused prompting guidance with real accuracy/cost outcomes.
- **OpenAI — Prompt Engineering Guide** — Vendor-official prompting best practices for content workflows.
- **HubSpot — AI Marketing Hub** — Practical framing of AI content strategy and tooling for marketers.

SEO & answer-engine optimization

- **Google Search Central** — The authoritative source for how Google crawls, indexes, and ranks.
- **Google Search Essentials** — The official rules a site must follow to appear in Google Search.
- **Ahrefs Blog** — Rigorous, data-driven SEO and content research.
- **Moz Blog** — Long-standing SEO authority; excellent on fundamentals.
- **Search Engine Journal** — Daily SEO/SEM news plus deep guides; growing AEO coverage.
- **Search Engine Land** — Breaking search-industry news and practitioner guides.

Analytics & measurement

- **Google Analytics Help (GA4)** — Official GA4 setup, reporting, and event/conversion documentation.
- **GA4 — Get Started** — Canonical GA4 onboarding article for new properties.
- **Google Analytics for Developers** — APIs for automated reporting and event ingestion.
- **Google — About Attribution** — Official primer on marketing attribution models.

Email & newsletter

- **Mailchimp — Email Marketing Guide** — Broad, practical strategy library from a leading ESP.
- **Litmus Blog** — Authoritative on email deliverability, rendering, and testing.
- **Really Good Emails** — Large curated gallery of real-world email design.
- **Campaign Monitor Resources** — Guides on newsletter design, segmentation, and automation flows.

Cadence & best-time-to-post

- **Sprout Social — Best Times to Post** — Large-dataset, annually-updated best-time-to-post research per platform.
- **Hootsuite — Best Time to Post** — Independent cadence/timing research to cross-check.
- **Buffer — How Often to Post** — Data-backed posting-frequency guidance per network.
- **Sprout Social — Social Media Analytics** — Framework for measuring what a posting strategy achieves.

Compliance & platform policy

- **FTC — Endorsements, Influencers & Reviews** — The FTC hub for disclosure rules governing paid/AI-assisted promotion.
- **FTC — Endorsement Guides FAQ** — Detailed FAQ on when/how to disclose material connections.
- **FTC — CAN-SPAM Compliance Guide** — The definitive rulebook for commercial email compliance.
- **GDPR — Official Overview** — Plain-language GDPR reference for marketers handling EU data.
- **Meta — Platform Terms** — The ToS governing automated use of Facebook/Instagram APIs.
- **X — Developer Agreement & Policy** — Rules for automation, rate limits, and acceptable use on X.

Troubleshooting & Recovery

Known gotchas with fixes, health checks, log locations, recovery runbooks, and a plain-English FAQ.

Known gotchas & fixes

SYMPTOM	CAUSE	FIX
Hub shows 502 on `/api`; backend crash-loops	The scheduler (Temporal) isn't reachable	Ensure the `temporal` container is up; it must start before Postiz settles.
Login bounces back to sign-in	Accessed over plain http	Always use `https://postiz.wholetech.com` — the cookie needs HTTPS.
Portal returns "database is locked" / 502	Two connections writing at once	Already fixed: state is committed before the audit log writes. Don't re-introduce a second open connection mid-write.
New subfolder 301-redirect loops	Missing nginx location for the path	Add an explicit `location ^~ /path/` block.
Files copied but site 404s	Permissions after a recursive copy	`chmod -R a+rX` the copied directory.
Posts stopped going to X	Daily cost cap reached, or X credit empty	Expected if capped; otherwise top up X credit. Bluesky keeps posting.
Feed looks stale	Nightly feed build didn't run/ship	Check `/var/log/social-feed.log`; re-run `gen-social-feed.py`.

Health checks

```
# hub services
ssh root@64.227.29.192 docker ps

# is the site up?
curl -I https://postiz.wholetech.com

# what published recently
ssh root@64.227.29.192 tail /var/log/portal-pub.log
```

Recovery runbooks

AFTER A SERVER REBOOT

The Docker services restart automatically; the cron jobs resume on schedule. Verify with the health checks above.

IF POSTING MISBEHAVES

Hit the kill switch, diagnose from the logs, then resume.

IF THE HUB IS LOST

Follow Backup & disaster recovery to rebuild from the nightly backup.

FAQ

Does it post without me? Yes — that's the point. You can switch any tenant to require approval if you'd rather review first.

Will it ever make up a fact? No. Missing facts produce a held post, never a guess.

What does it cost to run? $24/month for the hub, plus pennies for X if you post there. Everything else is free.

Can I manage clients on it right now? Yes — see Managing Clients. Each client is a tenant with their own accounts and settings.

How do I stop everything fast? The kill switch — one command.

Which networks are live today? X and Bluesky. Mastodon, LinkedIn, and Meta are ready to connect.

13

Glossary & A–Z Index

Plain-English definitions of every term in this manual, plus an alphabetical index of where to find things.

Glossary

AEO — Answer-Engine Optimization
Optimizing content so AI assistants cite it in their answers, not just so it ranks in search.

app password
A limited, revocable password some networks (Bluesky, Mastodon) issue so an app can post without your main login.

atproto / AT Protocol
The open protocol behind Bluesky.

cadence
How many posts a tenant keeps in flight — effectively the posting rate.

channel
A connected social account a tenant can post to (e.g. x, bluesky). The special channel “dry” logs without sending.

cron
The scheduler that runs each job at a set time.

deterministic template
A fixed sentence pattern that slots in verified facts — no AI call, no per-post cost, predictable output.

droplet
A cloud server (DigitalOcean’s term).

held post
A drafted post withheld because a required fact wasn’t verified in the source. The honesty guard in action.

honesty guard
The rule that a post is only written when every fact it needs is present in the source data.

kill switch
A single file (PAUSE) that freezes all publishing instantly.

magic link
A signed, expiring URL that lets a client open their approval queue without a password.

multi-tenant
One system serving many separate clients, each isolated with their own accounts and settings.

OAuth
The secure handshake that lets our hub post to your account without holding your password.

Postiz
The open-source app that actually posts to 30+ social networks from one place.

PDS
Personal Data Server — where a Bluesky account’s data lives.

tenant

One client (or our own network) in the system — a config block with accounts, content, and cadence.

Temporal
The scheduling engine Postiz relies on to time posts.

token (access token)
The credential, stored after connecting an account, that lets the hub publish on its behalf.

A–Z index

Jump straight to any topic:

White-label

X / Twitter billing

APPENDIX

Field Lessons

Distilled from the Instagram reels and posts shared during the build — the market signals that shaped and stress-tested this system. The full board, with links to every source, lives at wholereach.com/insta/.

These reels weren't just links — read together, they map what actually works in AI marketing and what's hype. Here's what we took from them (grounded in the notes we traded on each one).

What wins in AI marketing

- **Drafting is the easy part — the value is the last mile.** The real gap was never writing posts; it was publishing them and reading the results. This is why we close the social loop the same way we closed it for the website: **publish → measure**. — from: “A social media system” (Metricool)
- **Volume is a trap.** The “point it at a site and pump out AI articles” approach produces what the post's own comments called “AI slop” — and Google and the answer engines now detect it. A real quality bar beats volume. — from: “Autonomous AI SEO product” (AutoSEO)
- **The moat is economics + trust:** a flat subscription (not per-article API fees) so cost doesn't climb with scale, and a human “yes” before anything ships. — from: AutoSEO + Metricool

Read the hype critically

- **“Free Claude” usually isn't.** Running it “free” on Ollama means a weaker open model is driving behind Claude's interface — fine for tinkering, not production. Know what's actually under the hood. — from: “Claude and Instagram”
- **Branded “modes” aren't features.** “Content Creator Mode” isn't a real Claude feature — it's the creator's framing. Separate the marketing from the mechanism; the underlying prompts can still be gold. — from: “Prompts for your content creator”
- **The comments are data.** The most useful signal under the AutoSEO post was the audience itself calling it slop. — from: AutoSEO

Technical gotchas worth remembering

- **Grade the live site, not a blocked mirror.** The engine once scored a throttled/blocked page instead of the real site because the site was blocking automated requests. Always read the live site. — from: AutoSEO walkthrough
- **Hosted vs self-host is a real choice.** Hosted connectors (Metricool) install nothing — you authorize in the browser. We chose self-host for control and flat cost, but the convenience is

real. — from: Metricool

- **Real tools surfaced worth adopting:** Metricool (social execution + analytics) and Clay (verified lead sourcing). — from: Metricool + “interesting adapter”

What to build — the reels drew the map

- **Two halves of the business:** Content (make it) and Outbound (find the clients). The prompt reels covered content; the Clay reel covered client-finding. — from: content prompts + “interesting adapter”
- **Turn prompt packs into role playbooks.** The six content prompts — activate the role, 50 ideas from a niche, understand the audience, scroll-stopping hooks, structure the piece, repurpose one into many — became the Content role's playbook. — from: “Prompts for your content creator”
- **Repurpose one idea into many** (carousels, threads, emails, scripts) is the highest-leverage content move — and exactly what the Content role does. — from: content prompts

How the collaboration works

- **The stream is market radar.** A steady scan of what competitors ship. The winning pattern is honest triage: adopt the real (Metricool, Clay, the six prompts), discard the hype (AI slop, fake “modes,” “free” models) — and fold the keepers straight into the roadmap. — from: the whole stream