Habits — Multi-Tenant Habit Tracker (Caddy + Cloudflare Access + Redis)

A lightweight, multi-tenant habit tracker hosted on your Mac mini and exposed securely via **Cloudflare Tunnel** and **Cloudflare Access**. Each user's data is isolated by their Access email. Supports **daily logging with amounts**, **goal-based streaks**, **30-day heatmap**, and **goal progress bars**.

- Public URL: `https://habits.divyekant.com` (behind Cloudflare Access)
- Reverse proxy: **Caddy** → `habit-api:9300`
- Data store: **shared Redis** (same instance as mocks/webhook inbox), separate logical DB or namespaced keys
- Runtime: **Docker Compose** on external network `web_tunnel`

---

1) Architecture

```
Browser (CF Access SSO)
        │   CF-Access-Authenticated-User-Email
        ▼
Cloudflare Tunnel  (CNAME → Tunnel ID)
        ▼
Caddy (8080) ──────▶ reverse_proxy habit-api:9300
        ▼
Node/Express (Docker, 9300)
        ▼
Redis (shared container; namespaced keys)
```

Multi-tenancy

Tenant = the email in header `CF-Access-Authenticated-User-Email` injected by Cloudflare Access. All keys are prefixed with `ht:u:<uid>` (uid = lower-cased email). No data leakage across tenants.

---

2) Code Layout

```
/dev/apps/habits
├── server/
│   └── index.js           # API + static serving (multi-tenant, goals, stats,
export)
├── public/
│   ├── index.html         # Production UI (no dev headers)
│   └── app.js              # Calls API, renders list + stats/heatmap
├── Dockerfile
├── docker-compose.yml
└── README.md               # this file
```

---

3) Environment

Create `.env` in `/dev/apps/habits`:

```
REDIS_URL=redis://default:${REDIS_PASS}@redis-shared:6379/3
NODE_ENV=production
PORT=9300
# For production keep this disabled; dev only if you need curl without Access
ALLOW_X_USER=false
```

> You already run `redis-shared` and `web_tunnel`. We use DB `/3` for neat separation; keys are namespaced regardless.

---

4) Caddy & Cloudflare

Caddyfile (inside the running Caddy container)

Add this host block (already done during setup):

```
http://habits.divyekant.com {
    encode zstd gzip
    reverse_proxy http://habit-api:9300
}
```

Reload:

```
docker exec -it caddy caddy fmt --overwrite /etc/caddy/Caddyfile
docker exec -it caddy caddy reload --config /etc/caddy/Caddyfile
```

Cloudflare Tunnel

Route DNS by **Tunnel ID** (not the name):

```
cloudflared tunnel route dns <TUNNEL_ID> habits.divyekant.com
```

Cloudflare Access (Zero Trust → Access → Applications)

- **Type:** Self-hosted
- **Domain:** `https://habits.divyekant.com/*`
- **Policy:** Include your allowlist (e.g., `@clevertap.com` and/or personal email)
- The app injects header `CF-Access-Authenticated-User-Email`, which the API uses as the tenant id.

---

5) Build & Run

```
cd /dev/apps/habits
docker compose down
docker compose up -d --build
```

Sanity:

```
# API direct (container)
curl -s http://127.0.0.1:9300/health

# Host header via Caddy
curl -s -H "Host: habits.divyekant.com" http://127.0.0.1:8080/health
```

Open `https://habits.divyekant.com` in the browser → Cloudflare Access → UI loads.

---

6) Data Model (Redis)

Per-tenant (`uid = cf-access email, lower-cased`):

```
ht:u:<uid>:index             SET      habit IDs
ht:u:<uid>:habit:<id>        HASH     { id, name, template, type, unit,
dailyGoal, createdAt, archived }
ht:u:<uid>:streak:<id>       HASH     { current, longest, lastDoneDate }
ht:u:<uid>:log:<id>:<date>   STRING   "1" if the day is completed (goal met)
ht:u:<uid>:amt:<id>:<date>   STRING   numeric amount logged for the day
ht:u:<uid>:rl:done:<id>:<minute> STRING per-minute rate limit counter (TTL 70s)
```

Completion: A day is "complete" if `dailyGoal > 0` and `amount >= dailyGoal`, else if `amount > 0`.
Streak: increments only on *new* completions; consecutive days detected by `lastDoneDate`.

---

7) API Reference (v1)

All routes require Cloudflare Access (no dev headers in prod).

Health

- `GET /health` → `{ ok: true }`
- `GET /health/redis-roundtrip` → `{ ok: true }` simple set/get probe

Identity

- `GET /api/me` → `{ uid }`

Habits

- `GET /api/habits` → `{ habits: [...] }`
- `POST /api/habits`
  Body:

```
{
  "name": "Read",
  "template": "",
  "type": "reading",
  "unit": "pages",
  "dailyGoal": 20
}
```

  → `{ id, name, template, type, unit, dailyGoal }`

- `PATCH /api/habits/:id`
  Body (any subset):

```
{ "name": "Evening Reading", "archived": "1", "type":"reading",
"unit":"pages", "dailyGoal":25 }
```

  → `{ ok: true }`

- `DELETE /api/habits/:id` → `{ ok: true }`
  (Soft delete: removes habit + streak objects from index; logs/amounts are not iteratively purged)

Logging & Streaks

- `POST /api/habits/:id/done`
  Body:

```
{ "date": "2025-11-11", "amount": 22 }
```

→ `{ ok, date, amount, completed, current, longest, goal }`
Notes:
- Goal met → marks day complete and updates streak.
- Rate limit: **10 marks/min per habit per user** → `429 {"error":"slow_down"}`.

Stats & Export

- `GET /api/stats` → per habit:

```
{
  "id":"abc123",
  "name":"Read",
  "type":"reading",
  "unit":"pages",
  "dailyGoal":20,
  "current":5,
  "longest":12,
  "successRate30": 70,
  "progress30": { "total": 430, "target": 600, "pct": 72 },
  "last30":[ { "date":"2025-10-13","done":true,"amount":22 }, ... ]
}
```

- `GET /api/export?format=json|jsonl&attachment=1`
  Downloads user's habit definitions (not the per-day logs in this version).

---

8) Frontend (UI) Overview

- **Create** habits with `type / unit / dailyGoal`.
- **Log amount** daily (prompt) → if goal met, the day is "done" and streak increments.
- **Your habits** list: rename, edit goal, archive/unarchive, delete, log today.
- **Overview**: 30-day completion heatmap + current/longest streaks + 30-day goal progress bar.

The UI is **production-clean** (no dev headers). Authentication relies solely on Cloudflare Access.

9) Ops & Maintenance

Update the app

```
cd /dev/apps/habits
docker compose up -d --build
```

View logs

```
docker logs --tail=200 -f habit-api
```

Quick Redis inspection

```
docker exec -e REDISCLI_AUTH=$REDIS_PASS -it redis-shared redis-cli -n 3 KEYS
'ht:*' | head
```

Caddy reload

```
docker exec -it caddy caddy fmt --overwrite /etc/caddy/Caddyfile
docker exec -it caddy caddy reload --config /etc/caddy/Caddyfile
```

10) Security

- **AuthN/AuthZ** via Cloudflare Access. The API trusts only `CF-Access-Authenticated-User-Email`.
- **Dev override** (`ALLOW_X_USER`) is **disabled** in prod. Only enable for localhost testing and set back to `false`.
- Rate limiting on `/done` to prevent abuse.
- No PII beyond email as tenant id and user-entered habit names.

11) Troubleshooting

Symptom	Likely Cause	Fix
`habits.divyekant.com` shows default homepage	Caddy vhost not matching	Ensure Caddyfile has `habits.divyekant.com { reverse_proxy http://habit-api:9300 }` and reload.

Symptom	Likely Cause	Fix
`401 unauthorized` in UI	No CF Access header	Sign in via Cloudflare Access, ensure app policy includes your email.
`429 slow_down` on logging	Rate limit triggered	Wait a minute; designed to avoid spam logging.
Health OK direct but not via host header	Caddy not reloaded or hostname typo	`curl -H "Host: habits.divyekant.com" http://127.0.0.1:8080/health`, then fix Caddyfile + reload.
Stats empty but habits exist	No logs yet	Log at least once per habit.

---

12) Backups & Data Retention

- Data is in **Redis** (`redis-shared`). If Redis is ephemeral, configure a volume or RDB/AOF persistence (your main redis already uses `volumes: [redis_data:/data]`).
- Optional: nightly export of habit metadata and (future) logs to a file (see roadmap).

---

13) Versioning

- API v1 (this README). Backward-compatible additions should avoid breaking route shapes.
- Keep changelog in this README or a `/CHANGELOG.md`.

Next Steps (Prioritized Roadmap)

P0 — Reliability & Hygiene

1. **Redis persistence checks**: Confirm AOF/RDB enabled for `redis-shared` and retention policy is acceptable.
2. **Access groups**: Add Access Group (e.g., “Personal”) instead of hardcoded email list.
3. **Disable dev overrides**: Verify `ALLOW_X_USER=false` in prod image/environment.

P1 — Product polish

1. **PWA install + offline**: Add manifest + service worker so users can pin the app and log offline (queue and sync).
2. **Calendar view per habit**: Drill-down with monthly calendar + edit past days (amount corrections).
3. **Charts**:
4. Reading: pages/day trend, time to finish current book (ETA = remaining / avg pace).
5. Weight: line chart (kg) + 7-day moving average; show deficit adherence.

6. Workout: minutes/week, intensity buckets.
7. **Templates "with smarts"**: choosing a template pre-fills `type/unit/goal` and shows context tips (e.g., "20 pages/day → 600/mo").

P2 — Reminders & Integrations

1. **Daily reminder** (local browser notifications). Optional: email/SMS via Postmark (when you're ready).
2. **Webhook** on mark-done (for automations, e.g., Cronicle/Home Assistant).
3. **Import/Export v2**: include per-day logs (NDJSON), and a restore endpoint.

P3 — Collaboration & Social

1. **Share read-only streak page** (signed URL with TTL).
2. **Friends view**: compare completion rates and streaks within an Access group.

P4 — Admin & Ops

1. **Metrics**: Prometheus/Grafana or Datadog counters for requests, errors, RL, Redis timings.
2. **Audit log**: admin-only route to view actions per user.
3. **E2E tests**: Playwright/Cypress suite for create → log → stats.

P5 — Stretch

1. **Habit formulas**: derived habits (e.g., "hydrate 2L" split into 4 × 500ml tasks).
2. **Streak freeze**: one "skip day" per month to protect long streaks.
3. **Mobile-first redesign**: optimize tap targets and input flows for one-hand logging.

Quick Tasks You Can Do Next (5–10 mins each)

- Add **PWA boilerplate**: `manifest.json`, `service-worker.js` (cache `/public/*`), prompt to install.
- Add **per-habit detail page**: `/h/:id` that renders last 90 days, total progress, log table.
- Add **book progress** sample: store per-habit metadata like `totalPages`, `currentPage`; derive ETA.