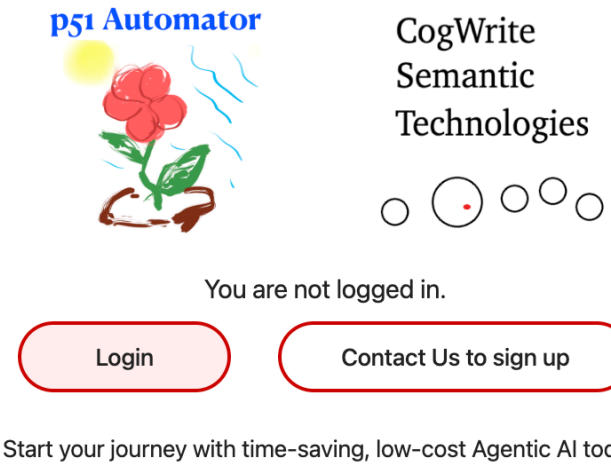
p51-local-automator — Architecture and Overview

Date: 2026-05-27. Application version 2026.05.27.1.



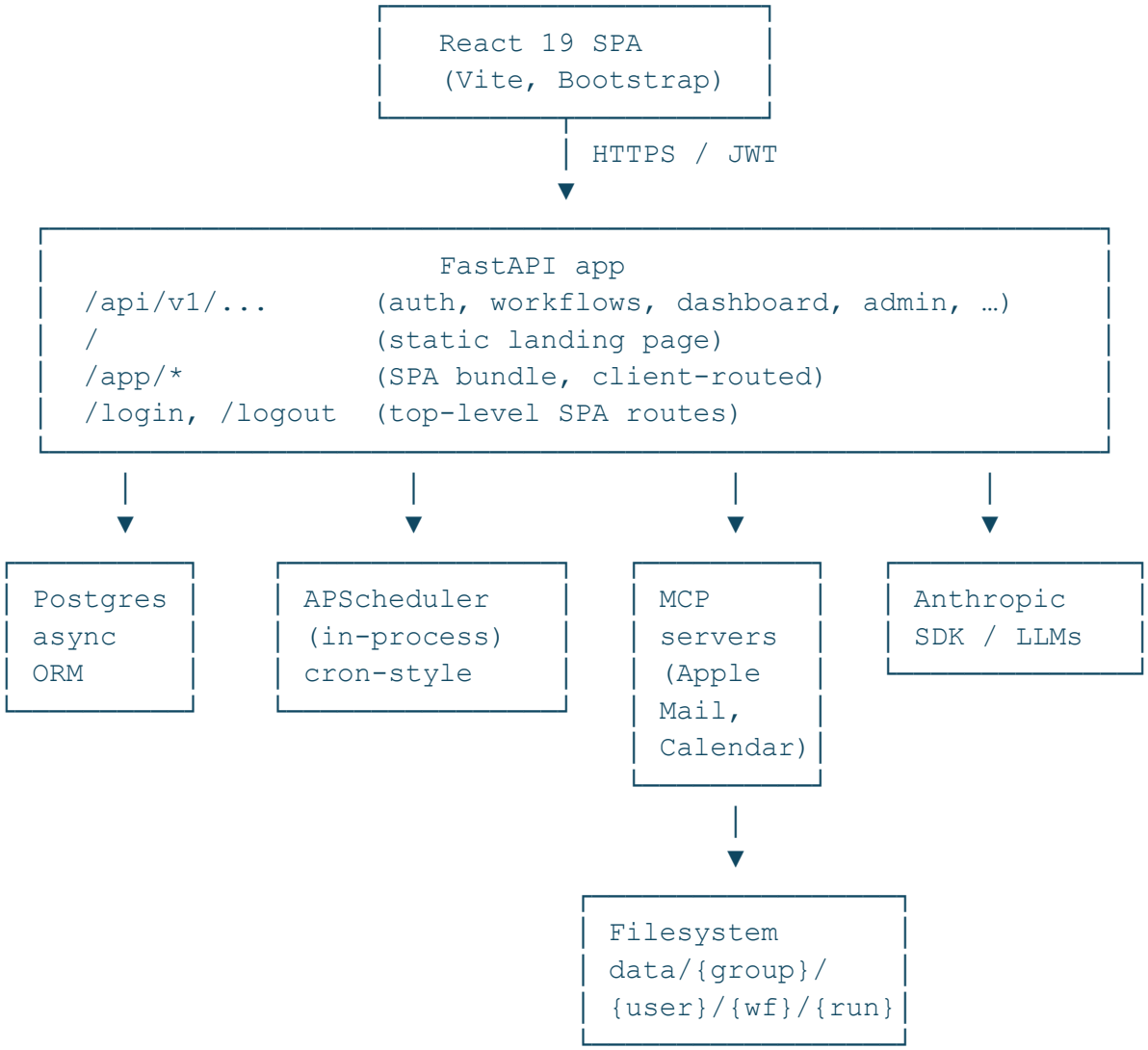
A. 1. Overview

p51-local-automator (internally "p51") is a Mac-based server platform that lets small businesses and individuals build, schedule, and run AI-assisted automations against their own email, calendars, files, and databases. Typical automations include:

- Monitoring an inbox for messages on user-defined topics and producing a categorized report.
- Generating a daily calendar digest with prep notes.
- Running parameterized SQL queries against operational databases and getting an LLM narrative on the results.
- Drafting or sending auto-replies to inbound email under user-defined rules.
- Running an agentic pipeline against a data collection and producing a multi-stage report.

All processing happens locally on the user's Mac (desktop or Mac Mini server). The only outbound calls are to the LLM API and to the user's own Google services when those are configured. There is no SaaS dependency for storage, scheduling, or orchestration.

B. 2. System Architecture



Key behaviors:

- A single uvicorn process hosts both the API and the SPA bundle (production) or proxies to Vite (development).
- APScheduler runs in-process, polling and firing scheduled workflow runs through the same code path as manual triggers.
- LLM calls go directly to Anthropic via the official SDK (no CrewAI or middleware).
- Apple Mail and Apple Calendar are accessed through local MCP servers; Google services use the Google APIs directly (Workspace OAuth or, for consumer Gmail, IMAP/SMTP with App Passwords).

- Generated artifacts live under a per-group/per-user/per-workflow/per-run folder tree on the host filesystem. Self-describing metadata is embedded in every artifact (JSON __meta__, Markdown frontmatter, Excel Provenance sheet, CSV #-comment lines, PNG matplotlib footer).

C. 3. Tech Stack

Layer	Choice
Backend framework	FastAPI
ORM	SQLAlchemy 2.x (async) over asyncpg
Migrations	Alembic
Database	PostgreSQL
Authentication	fastapi-users with JWT (7-day tokens)
Scheduler	APScheduler (in-process)
LLM	Anthropic Python SDK (Claude)
Apple integrations	Local MCP servers + osascript (AppleScript)
Google integrations	google-api-python-client (OAuth) + stdlib imaplib/smtplib (consumer Gmail)
Frontend framework	React 19 + TypeScript
Build tool	Vite
UI library	react-bootstrap
State	Zustand
Versioning scheme	CalVer (YYYY.MM.DD.N) — single source of truth in backend/__init__.py

Python 3.12+. Node 22+. No Docker.

D. 4. Multi-tenancy, Roles, and Admin Functions

1. Solo vs team use

The same code supports two deployment modes; the difference is how many human users share a p51 instance:

- **Solo mode** (a single human on their own Mac or a private Mac Mini): one user account with full access to their own workflows. Group-level controls still exist but are effectively no-ops.

- **Team mode** (a Mac Mini serving multiple humans): each human gets their own p51 user account inside a group. Group-level admins manage members and per-group defaults; a superuser manages groups and system-wide defaults.

2. Data model

Two hierarchical tenancy keys appear on every workflow and related row:

- group_id — the team a workflow belongs to. All multi-tenant queries filter by group.
- user_id — the individual within that group who owns the workflow.

Every list and detail endpoint applies a role-scoped where-clause (_run_scope_filter in backend/api/dashboard.py):

Role	Visibility
Employee	Only their own workflows / runs
Manager / Group admin	All workflows / runs in their group
Superuser	All workflows / runs system-wide

3. Roles

Four roles, in order of authority:

1. **Employee** — the default. Can create, configure, run, schedule, and delete their own workflows. Can view their own past runs and artifacts. Can configure their own Profile (outbound results email, login email, etc.).
2. **Manager** — same as employee, plus visibility into all workflows in their group. Useful for oversight without administrative authority.
3. **Group admin** — same as manager, plus the ability to manage users in their group (add, disable, change role within the group) and edit group-level settings.
4. **Superuser** — system-wide authority. Creates and disables groups, edits system defaults, runs maintenance sweeps, and edits the workflow type catalog.

Role assignments live as boolean columns on the api_users row (is_manager, is_groupadmin, is_superuser). The login email is User.email (managed by fastapi-users) and is independent of any email account used by workflows or for delivery of results.

4. Admin functions

Group-admin / manager scope: - Manage Users (/app/admin/users) — add/edit/disable users in their group. - Group Settings (/app/admin/group-settings) — override system defaults for a single group (e.g., per-group cap on emails fetched per run).

Superuser scope: - All group-admin pages, system-wide. - Manage Groups (/app/admin/groups) — create/edit/disable groups. - Workflow Categories (/app/admin/workflow-categories) — edit category metadata that drives the Dashboard grid. - Workflow Types (/app/admin/workflow-types) — edit per-type metadata (name, description, default config, schedulable / emailable flags). - API Settings (/app/admin/settings) — system-wide tuning (limits, file_system_root, token budgets). - Maintenance (/app/admin/maintenance) — archive sweep, purge of soft-deleted rows, run-history retention.

5. Settings resolution chain

Numeric workflow limits resolve through a three-layer chain (resolve_int_setting() in backend/services/workflow_engine.py):

workflow.config[<key>] → group_settings(group_id, <key>) → api_settings(<key>) → runner fallback

Hardcoded absolute ceilings (e.g. ABS_MAX_AGENT_TURNS=100, ABS_MAX_LLM_TOKENS=16384) cap runaway costs and cannot be overridden through settings.

E. 5. Workflow Types Supported

Seven standard types ship today, plus one ad-hoc workflow option.

ID	Name	What it does	Schedulable	Emailable Results
1	Email Topic Monitor	Reads recent messages from an Apple Mail / Workspace Gmail / consumer Gmail inbox and categorizes them against user-defined topics.	✓	✓
2	Transaction Data Analyzer	Loads a user-provided data file, computes profile statistics, generates charts, and produces an LLM narrative summary.	✓	✓
3	Calendar Digest	Pulls upcoming events from Apple Calendar or Google Calendar and produces a Markdown digest with prep notes.	✓	✓
4	SQL Query Runner	Runs a read-only query against a user-configured database connection. Returns CSV + Excel + an LLM narrative on the result rows.	✓	✓

ID	Name	What it does	Schedulable	Emailable Results
5	Auto-Reply (Draft Only)	Watches an inbox; for each message matching user rules, drafts a reply and saves it to the Drafts folder for human review.	✓	✗ (output is email)
6	Auto-Reply (Approve Before Send)	Same as Type 5 but pauses for human approval in the UI before sending. Sent via the source account.	✓	✗ (output is email)
7	Analyze Data Collection (AWF-1)	Multi-stage agentic pipeline against a data collection. Produces stage-by-stage charts and a final analyst report.	✗ (too expensive to cron)	✓

1. Ad-hoc workflows

A separate, lightweight surface for iteration. Currently one option:

- **Ad-hoc Email Topic Monitor** — one re-configurable instance per user. Useful for trying different topic combinations, scopes, and time windows before promoting a tuned configuration to a saved Type 1 workflow. Ad-hoc workflows cannot be scheduled.

The Dashboard now surfaces both lists side-by-side: "Standard Workflow Types" (the seven cards) above "Ad-hoc Workflow Options" (the full-width list).

2. Self-describing artifacts

Every artifact a workflow run produces carries an embedded provenance block identifying the run, its subject, and the workflow that produced it. See backend/services/artifact_meta.py for the build/wrap functions; readers should consult the per-format read conventions documented in CLAUDE.md (e.g., pd.read_excel(path, sheet_name="Results") to skip the auto-prepended Provenance sheet).

F. 6. Optional Email Delivery of Results

Any workflow type with emailable_results=true (currently 1, 2, 3, 4, 7) can be configured to email its outputs to the workflow owner after a successful run. The flow:

1. **Per-user designation (Profile page):** the user picks ONE outbound account from three supported services — Apple Mail (via the Mail.app client on this Mac), a connected Workspace Gmail (OAuth), or a consumer Gmail (App Password / SMTP). Stored as outbound_service + outbound_identifier on api_users.
2. **Per-workflow opt-in (Configure form):** a section "Email results" appears for emailable types. Toggle ON, then check the specific artifact kinds (e.g., "Excel summary report", "Chart images (PNG)") to attach.

3. **Run hook:** after the runner completes and the run is marked completed, send_results_email() resolves attachments by regex against artifact basenames, composes a fixed-template message, dispatches via the user's chosen backend, and writes one row to workflow_run_email_log. The run's status is unaffected by delivery outcome — delivery failures surface as a red envelope badge on the run row but do not fail the workflow.

Sender / destination semantics:

- Apple Mail: the destination is an explicit email address; the sender is Mail.app's default account (or, in an "Advanced" disclosure, a specific Mail.app account the user names — useful for shared-sender / KPI-bot patterns).
- Workspace Gmail OAuth: self-send (the connected account's own address) using the existing gmail.send scope.
- Consumer Gmail (App Password): self-send via smtp.gmail.com:465 using stdlib smtplib.

G. 7. Backend (API Routes and Auth)

All API routes are mounted under /api/v1. Authentication is JWT bearer (7-day tokens) issued by fastapi-users at /api/v1/auth/jwt/login.

1. Auth

Route	**Purpose**
POST /api/v1/auth/jwt/login	Username + password → JWT
POST /api/v1/auth/jwt/logout	Invalidate session
POST /api/v1/auth/register	Self-registration (disabled by default in team mode)
GET /api/v1/users/me	Current user info + outbound email prefs
PATCH /api/v1/users/me	Update outbound email prefs

2. Workflows

Route	**Purpose**
GET /api/v1/workflows	List user-visible workflows (scoped by role)
POST /api/v1/workflows	Create a workflow
GET /api/v1/workflows/{id}	Detail
PUT /api/v1/workflows/{id}	Update config / schedule / name / enabled
DELETE /api/v1/workflows/{id}	Soft delete

Route	Purpose
POST /api/v1/workflows/{id}/run	Manual trigger
GET /api/v1/workflows/{id}/runs	Run history (with email-send badges)
POST /api/v1/workflows/bulk-delete	Bulk soft-delete
GET /api/v1/workflow-types	Catalog (drives Dashboard cards + Configure form)
GET /api/v1/workflow-categories	Category catalog
GET /api/v1/schedules	All schedulable workflows + their schedules
GET /api/v1/runs/{run_id}	Single run detail + email-send status
GET /api/v1/runs/{run_id}/steps	Per-step status / output summary
GET /api/v1/runs/{run_id}/artifacts	List artifacts for a run
POST /api/v1/pending-replies/{id}/approve	Type 6 approve flow
POST /api/v1/pending-replies/{id}/reject	Type 6 reject flow
POST /api/v1/pending-replies/{id}/save-draft	Type 6 save-as-draft flow

3. Ad-hoc

Route	Purpose
GET /api/v1/ad-hoc/email-topic-monitor	Auto-create or fetch the user's ad-hoc Email Topic Monitor row
PUT /api/v1/ad-hoc/email-topic-monitor	Save configuration to the ad-hoc row
POST /api/v1/ad-hoc/email-topic-monitor/run	Run the ad-hoc workflow
POST /api/v1/ad-hoc/email-topic-monitor/test	Test inbox access for the configured accounts
POST /api/v1/ad-hoc/email-topic-monitor/clear	Clear the ad-hoc configuration
GET /api/v1/ad-hoc/runs	Cross-type ad-hoc run history

4. Connections (Google)

Route	Purpose
POST /api/v1/gmail/oauth/start	Begin Workspace Gmail OAuth

Route	Purpose
GET /api/v1/gmail/oauth/callback	OAuth callback target
GET /api/v1/gmail/accounts	List the user's connected Workspace Gmail accounts
DELETE /api/v1/gmail/accounts/{id}	Revoke a connection
GET /api/v1/google-calendar/calendars	List Google calendars on a connected account

5. Dashboard / Files / Artifacts / System

Route	Purpose
GET /api/v1/dashboard/stats	Top-of-Dashboard counters
GET /api/v1/dashboard/recent-runs	Recent-runs table on Dashboard
GET /api/v1/files/list	Browse files under the per-user input sandbox
GET /api/v1/artifacts/{id}/download	Download a specific artifact file
GET /api/v1/system/version	App + DB-revision report
GET /api/v1/system/health	Health probe

6. Admin (role-gated)

Route	Purpose	Min role
GET/POST/PUT /api/v1/manage/users	User CRUD	Group admin (own group) / superuser (any)
GET/POST/PUT /api/v1/manage/groups	Group CRUD	Superuser
GET/PUT/DELETE /api/v1/group-settings/{name}	Per-group setting overrides	Group admin
GET/PUT/DELETE /api/v1/settings/{name}	System-wide setting	Superuser
GET /api/v1/settings/webapp_options	UI feature flags	Any authenticated
POST /api/v1/settings/validate-path	Probe a path's writability	Superuser
GET /api/v1/admin/workflow-categories, PATCH .../{id}	Edit category catalog	Superuser

Route	Purpose	Min role
GET /api/v1/admin/workflow-types, PATCH .../{id}	Edit type catalog	Superuser
POST /api/v1/scheduler/start, POST .../stop, GET .../status	Control APScheduler	Superuser
GET/POST /api/v1/maintenance/*	Archive, purge, history	Superuser

H. 8. Frontend (Pages and URL Routes)

Two URL spaces:

- / — splash page (backend/landing/index.html) served by FastAPI. Static marketing-style landing; the only HTTP path outside the SPA. Links to /app for authenticated entry and /login for sign-in.
- /app/* — the SPA. All client-routed by React Router.

1. SPA routes (under /app)

Path	Page	Description
/app	Dashboard	Stats, recent runs, Standard Workflow Types grid, Ad-hoc Workflow Options list
/app/workflows	Workflows	List/create/configure/delete user workflows
/app/workflows/:id	WorkflowDetail	Edit, schedule, run, history
/app/workflows/:id/pending-replies	PendingReplies	Type 6 approval queue
/app/runs/:runId	RunDetail	Steps + artifacts for a single run
/app/connections	Connections	Connect/revoke Workspace Gmail accounts
/app/schedules	Schedules	All schedulable workflows + their cron triggers
/app/ad-hoc/email-topic-monitor	AdHocEmailMonitor	Run/edit the ad-hoc Email Topic Monitor
/app/ad-hoc/runs	AdHocRuns	Cross-type history of ad-hoc runs

Path	Page	Description
/app/profile	Profile	Account info + outbound results email config
/app/admin/users	ManageUsers	Group admin / superuser
/app/admin/groups	ManageGroups	Superuser
/app/admin/workflow-categories	ManageWorkflowCategories	Superuser catalog edits
/app/admin/workflow-types	ManageWorkflowTypes	Superuser catalog edits
/app/admin/settings	Settings	System-wide
/app/admin/group-settings	GroupSettings	Group admin
/app/admin/maintenance	Maintenance	Superuser
/app/logout	Logout	Clear token, redirect to splash

2. Top-level pages (outside /app)

Path	Page	Description
/	Splash	Landing page served by FastAPI (backend/landing/index.html)
/login	Login	Sign-in form
/logout	Logout	Convenience top-level route

3. Navigation surfaces

- Top nav bar: Dashboard, Workflows, Ad-hoc Workflows dropdown, Schedules, Connections.
- User dropdown (top-right): username/email/group display, Profile, Logout.
- Admin entries: currently scattered (Manage Users, Settings, etc. each as their own top-level link visible to authorized roles). A consolidated "Administration" dropdown is a queued reorg.

I. 9. Deployment — Desktop (Single User)

Target: one human on their own Mac (laptop or always-on desktop).

Characteristics: - Apple Mail / Apple Calendar work freely — they're the user's own Mail.app, with all of their personal accounts, and only that human reads/writes them. - The user can use any outbound delivery service (Apple Mail, Workspace Gmail, consumer Gmail) — all three

behave identically. - Group functions are vestigial: one user, one group, no group-level decisions to make.

Setup: 1. Install Python 3.12+, Node 22+, PostgreSQL. 2. git clone …, cd p51-local-automator, python3 -m venv .venv && source .venv/bin/activate, pip install -r backend/requirements.txt. 3. Create the database: createdb p51_automator. 4. Apply schema: alembic upgrade head. 5. Start backend: python3 -m uvicorn backend.main:app --reload --port 8000. 6. Start frontend dev server: cd frontend && npm install && npm run dev (or build with npm run build and let uvicorn serve the bundle). 7. Optional: configure a launchd LaunchAgent so uvicorn auto-starts on login — useful if you want the Mac to wake up and run morning workflows before you log in.

There's no team-mode configuration to enable; the platform behaves correctly with a single user account in the default group.

J. 10. Deployment — Shared Mac Mini or Remote Server (Team)

Target: a Mac Mini acting as a team server, with multiple human users sharing it. Each user signs in to p51 over the local network via their browser; the Mac Mini doesn't need a connected display in normal operation.

Characteristics and constraints:

- Apple Mail on the Mac Mini is single-tenant by macOS design: the macOS user that runs uvicorn (typically cogmgr in p51's canonical setup) owns one Mail.app configuration. Every p51 user can read whatever's in that Mail.app. Configure only shared mailboxes there (e.g., acme_accounts_payable@…, info@…, support@…) — never personal mail.
- Personal email belongs in each user's own Gmail (Workspace or consumer), connected per-p51-user via OAuth (Workspace) or App Password (consumer). These are properly multi-tenant: each user's credentials and tokens are scoped to their User row.
- For the email-delivery-of-results feature, the same model applies: each user's Profile selects their personal outbound service. Apple Mail-as-outbound is available but should generally be reserved for shared-sender patterns (e.g., a KPI bot account configured in the Mac Mini's Mail.app).
- Future Google Workspace CLI-based workflows will require Workspace accounts specifically — when those ship, consumer-Gmail-only users won't be eligible to run them. Onboarding guidance for new team deployments should steer toward Workspace from the start unless there's a specific reason not to.

Setup: 1. Install macOS user account for the p51 backend (e.g., cogmgr); set up auto-login or fast user switching as appropriate. 2. Install Python / Node / PostgreSQL on that user account. 3. Same install steps as desktop (steps 1–6 above). 4. Run uvicorn as a launchd LaunchAgent under the cogmgr user account. Document the Automation (TCC) permissions Mail.app needs the first time AppleScript-driven send/read fires — those have to be granted interactively while logged in

at the Mac Mini's console. 5. Create the first superuser via the admin script; that superuser then creates additional groups and group admins via the Admin pages. 6. Each end user receives a login (email + temp password) from their group admin and signs in via http://<mac-mini>:8000/login from their own machine.

Concurrency / sharing notes:

- All users share the same uvicorn process and Postgres database — there is no per-user backend isolation. Multi-tenancy is enforced at the query layer via group_id / user_id filtering on every list/detail endpoint.
- Multiple users can run workflows simultaneously; per-workflow run locks prevent two concurrent runs of the same workflow but allow parallel runs of different workflows.
- The scheduler runs in-process; one APScheduler instance fires every scheduled workflow regardless of owner.

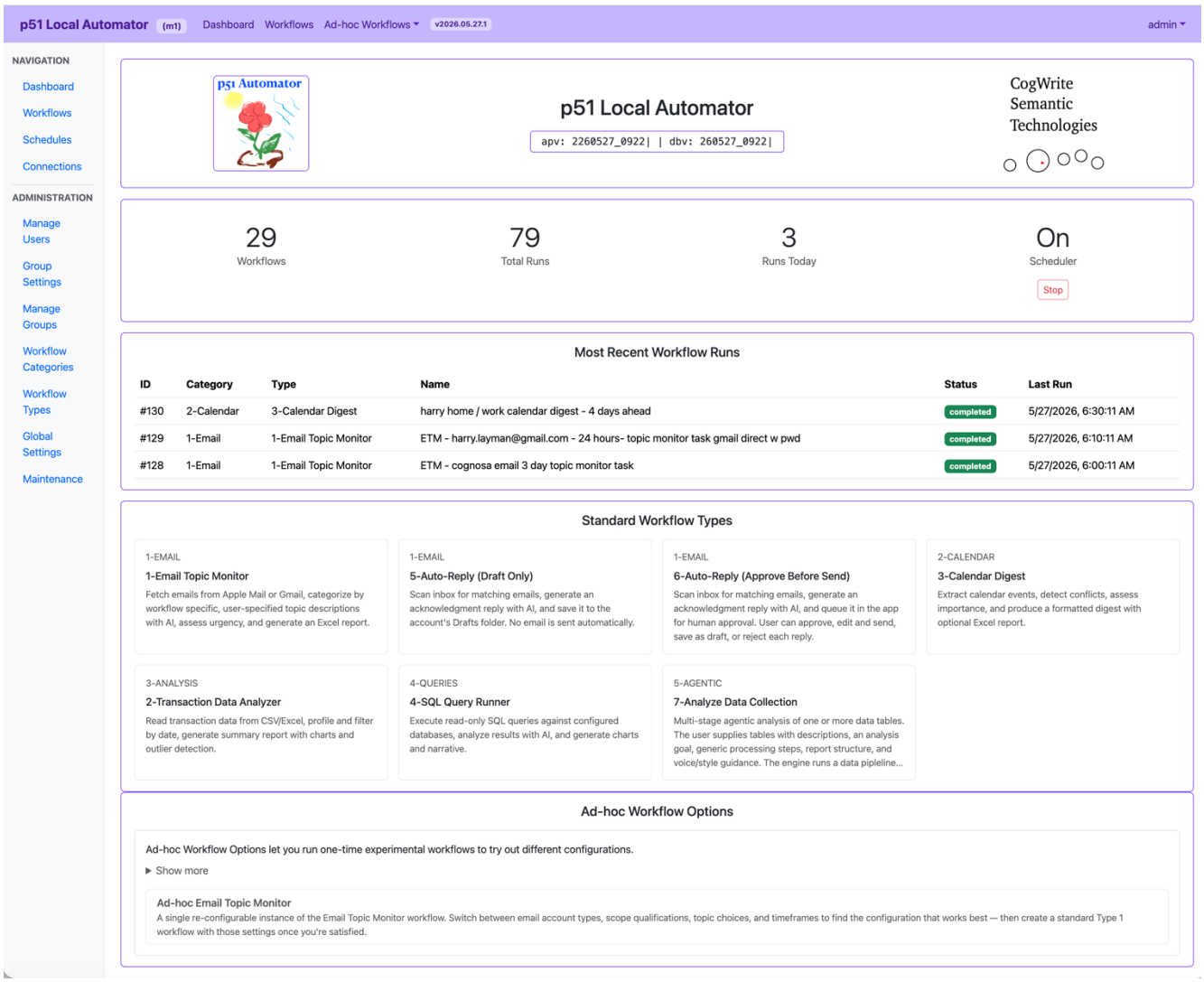
Three-instance multi-tenancy pattern (advanced):

If a team strictly needs per-user Apple Mail isolation on a single Mac Mini, an unusual but supported pattern is to run multiple independent p51 instances under different macOS user accounts on the same machine (different ports, different databases, different home directories). This gives each user their own Mail.app, their own MCP servers, and their own file_system_root, but only ONE of those macOS users can have an active GUI session at a time — so reliable Apple Mail integration only works for whichever user is currently logged in at the console. This pattern is documented for completeness; the recommended setup for most teams remains a single shared p51 instance with personal mail handled via per-user Gmail.

K. See also

- CLAUDE.md — operator-facing reference for conventions, settings chains, and feature specifics.
- docs/p51_deployment_modes_macmini_260525.md — concise statement of the two deployment-mode rules for Mac Mini installs.
- backend/services/results_email.py — entry point and registry for the "email me my results" feature.
- backend/api/__init__.py — single source of truth for the routes table.

L. Dashboard


p51 Local Automator
(m1)
Dashboard
Workflows
Ad-hoc Workflows
v2026.05.27.1
admin
NAVIGATION
Dashboard
Workflows
Schedules
Connections
ADMINISTRATION
Manage Users
Group Settings
Manage Groups
Workflow Categories
Workflow Types
Global Settings
Maintenance
p51 Automator
p51 Local Automator
apv: 2260527_0922| | dbv: 260527_0922|
CogWrite Semantic Technologies
29
Workflows
79
Total Runs
3
Runs Today
On
Scheduler
Stop
Most Recent Workflow Runs
ID
Category
Type
Name
Status
Last Run
#130
2-Calendar
3-Calendar Digest
harry home / work calendar digest - 4 days ahead
completed
5/27/2026, 6:30:11 AM
#129
1-Email
1-Email Topic Monitor
ETM - harry.layman@gmail.com - 24 hours- topic monitor task gmail direct w pwd
completed
5/27/2026, 6:10:11 AM
#128
1-Email
1-Email Topic Monitor
ETM - cognosa email 3 day topic monitor task
completed
5/27/2026, 6:00:11 AM
Standard Workflow Types
1-EMAIL
1-Email Topic Monitor
Fetch emails from Apple Mail or Gmail, categorize by workflow specific, user-specified topic descriptions with AI, assess urgency, and generate an Excel report.
1-EMAIL
5-Auto-Reply (Draft Only)
Scan inbox for matching emails, generate an acknowledgment reply with AI, and save it to the account's Drafts folder. No email is sent automatically.
1-EMAIL
6-Auto-Reply (Approve Before Send)
Scan inbox for matching emails, generate an acknowledgment reply with AI, and queue it in the app for human approval. User can approve, edit and send, save as draft, or reject each reply.
2-CALENDAR
3-Calendar Digest
Extract calendar events, detect conflicts, assess importance, and produce a formatted digest with optional Excel report.
3-ANALYSIS
2-Transaction Data Analyzer
Read transaction data from CSV/Excel, profile and filter by date, generate summary report with charts and outlier detection.
4-QUERIES
4-SQL Query Runner
Execute read-only SQL queries against configured databases, analyze results with AI, and generate charts and narrative.
5-AGENTIC
7-Analyze Data Collection
Multi-stage agentic analysis of one or more data tables. The user supplies tables with descriptions, an analysis goal, generic processing steps, report structure, and voice/style guidance. The engine runs a data pipeline...
Ad-hoc Workflow Options
Ad-hoc Workflow Options let you run one-time experimental workflows to try out different configurations.
Show more
Ad-hoc Email Topic Monitor
A single re-configurable instance of the Email Topic Monitor workflow. Switch between email account types, scope qualifications, topic choices, and timeframes to find the configuration that works best — then create a standard Type 1 workflow with those settings once you're satisfied.