BRAND AND DESIGN SYSTEM

go-schedule

Familiar scheduling, explicit behavior.

Cross-platform task scheduling with a CLI, desktop application, and background daemon.

A SHRUGGIETECH PROJECT · VERSION 1.0.0

FOUNDATION

Know the next run

People should recognize a scheduler immediately and understand what it will do. go-schedule keeps authored expressions visible, translates them into plain language, and surfaces the policies that change execution.

CATEGORY	Cross-platform task scheduling
ROLE	A readable scheduler for service-managed automation
AUDIENCE	Developers, operators, maintainers, and technical users who need exact behavior across operating systems
DESCRIPTOR	Cross-platform task scheduling with a CLI, desktop application, and background daemon
PARENT	ShruggieTech, with shared typography and an independent product identity

BRAND IDEA

Schedule intent becomes observable state

A task has an authored expression, a plain-language explanation, a next run when one exists, and an execution history. Event schedules state their lifecycle boundary without inventing a clock time.

Personality

- Precise and practical
- Calm under failure
- Transparent about lifecycle and policy
- Familiar to terminal and cron users

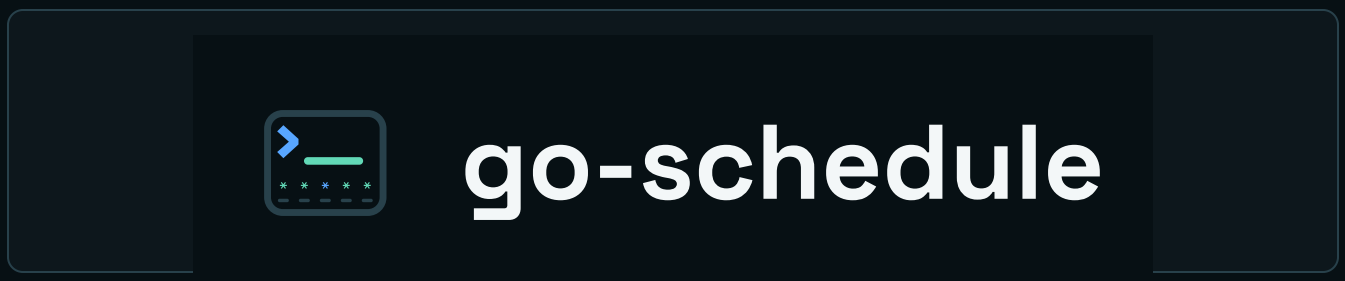
Promises

- Authored intent remains visible
- Explanation agrees with execution
- Policy appears when it changes behavior
- State never depends on color alone

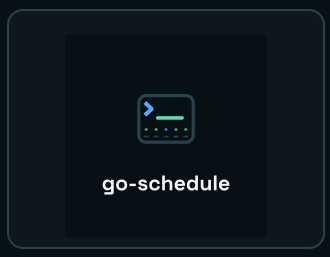
LOGO SYSTEM

Mark and lockups

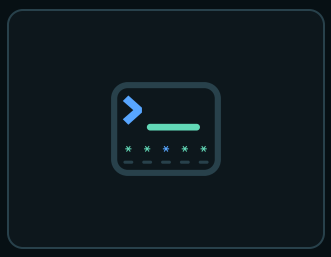
The mark combines a terminal prompt, a command line, and five schedule cells. The center Anchor Blue cell identifies an exact run point. Interval Mint cells represent recurrence.



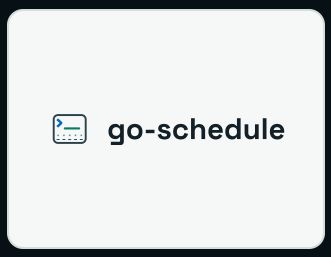
HORIZONTAL LOCKUP - HEADERS, DOCUMENTATION, REPOSITORY ARTWORK



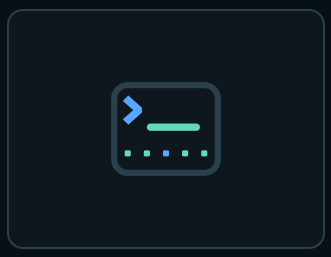
STACKED · 104 PX



FULL MARK · 36 PX



LIGHT SURFACE

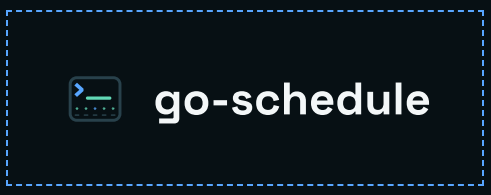


REDUCED · ≤ 32 PX

CLEAR SPACE

32 units around the artwork

Keep text, borders, icons, and crops outside the boundary.



PORTABLE MASTERS

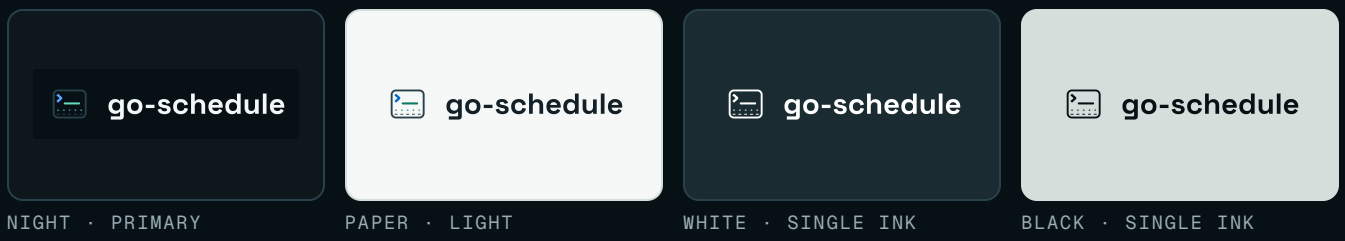
The wordmark and schedule-cell asterisks are outlined paths. No distributable logo depends on an installed font.

ASSET	MINIMUM
Reduced mark	16 px
Full mark	36 px
Horizontal	180 px
Stacked	104 px
Wordmark	120 px

LOGO SYSTEM

Backgrounds and use

Primary artwork uses Interval Mint and Anchor Blue on Night. Light surfaces use the deeper accessible accents. White and black variants support single-ink reproduction.



NIGHT · PRIMARY | PAPER · LIGHT | WHITE · SINGLE INK | BLACK · SINGLE INK

USE

- Use transparent variants when the destination already supplies a surface.
- Use SVG wherever supported.
- Keep the supplied Night background in tiny system icons.
- Use the reduced mark at 32 px and below.
- Preserve full clear space.

AVOID

- Rotation, skew, stretch, bevel, outline, or glow.
- Live-text substitutes for the wordmark.
- A nearly matching square tile placed on another background.
- Recolored individual schedule cells.
- Decorative clocks, bells, speed lines, or calendar pages.
- A combined go-schedule and ShruggieTech logo.

COVER COMPOSITION

The mark stands alone

The guide cover uses the transparent mark directly on Night. The page title carries the product name and the supporting copy carries the descriptor. This removes the mismatched square and avoids repeating the same information inside a hero image.

COLOR

Scheduling palette

Neutral surfaces carry most of the interface. Mint describes recurrence and ready state. Blue marks exact points, links, and focus. Amber requests attention. Red marks failure or destructive action.

Interval Mint
#62D9B7
11.09:1 on Night

Anchor Blue
#58A6FF
7.60:1 on Night

Hold Amber
#F2B84B
10.73:1 on Night

Stop Red
#E05F5F
5.46:1 on Night

Night
#071014

Panel
#0D171C

Raised
#13232A

Line
#28414B

LIGHT READING SURFACE

TOKEN	HEX	CONTRAST
Ink	#132027	15.58:1
Light Muted	#4A616B	6.13:1
Light Interval	#087A62	4.96:1
Light Anchor	#0067C5	5.26:1
Light Hold	#805500	6.12:1
Light Stop	#B4232A	6.12:1

HARD RULE

Interval Mint measures **1.62:1** on Paper. It cannot carry text there. Use Light Interval and the corresponding light variants for essential text.

Every contrast number is recalculated from the token file during verification.

TYPOGRAPHY

Display, reading, technical detail

SPACE GROTESK 700

Know the next run.

Headings and short display statements.

GEIST 400 / 500

Readable schedules are useful when their execution policy is equally visible. Body and interface text should remain direct and calm.

Body copy, controls, descriptions, and documentation.

GEIST MONO 400

```
0 15 * * 1-5
@reboot
at 09:30 every weekday
2026-08-30T12:00:00Z  America/New_York
```

Schedule expressions, commands, paths, timestamps, logs, and metadata. Ligatures remain disabled.

FUNCTION	FAMILY	AVAILABLE WEIGHT
Display	Space Grotesk	500, 700
Body	Geist	400, 500
Technical	Geist Mono	400
Wordmark	Outlined artwork	Fixed vector

Use color, spacing, or a label for emphasis in technical text. Geist Mono has no bold face in this kit.

VISUAL LANGUAGE

Structure that explains behavior

Visual references come from cron fields, terminal prompts, timelines, task queues, service status, and run history. Every line or point should explain a relationship or a state.

RAILS

Sequence

Use aligned rails for recurring fields, history, and execution flow.

ANCHORS

Exact points

Blue points identify a selected or exact run boundary.

POLICY

Attention

Amber calls out catch-up, overlap, pause, or pending behavior.

STARTER ICONS

Icons use a 24-unit grid and a 1.5-unit stroke. Extend the set with direct scheduling, task, service, and history symbols.

SURFACES

- 1 px Line borders
- 4 to 8 px radii
- Minimal shadows
- Neutral large fills
- Small amber and red regions

MOTION

- 120 to 240 ms
- `cubic-bezier(.2,.6,.2,1)`
- State transitions and refreshes
- Reduced-motion support
- No decorative pulsing

VOICE AND WRITING

Explain the runbook

Use active voice, exact nouns, and observable outcomes. State prerequisites before commands. Name timezone, catch-up, overlap, and lifecycle boundaries whenever they change the result.

PREFER	AVOID
task, schedule, expression, run	automation magic, workflow wizard
daemon, service, startup, reload	always-on intelligence
timezone, catch-up, overlap, policy	set it and forget it
next run unavailable	never miss a beat
supported, experimental, deferred	flawless, effortless, revolutionary

SCHEDULE

At 09:30 every weekday in America/New_York.

LIFECYCLE

Runs once when the scheduler daemon starts. Reloading the task list does not run it again.

EMPTY STATE

No tasks match the current filters.

ERROR

The task could not start because its working directory is unavailable.

TERMINOLOGY

The product name is always lowercase: **go-schedule**. Use `daemon start` for the process lifecycle boundary. Use `host boot` only for operating-system startup behavior.

UI SYSTEM

Tokens and components

The kit includes a small implementation layer so the written rules can be tested in a real interface.

MORNING SYNC

At 09:00 every weekday

READY

`0 9 * * 1-5`

Next run: 2026-08-31 09:00 America/New_York

READY

Runnable and scheduled.

PENDING

Waiting for a lifecycle boundary.

FAILED

Execution ended in failure.

FILE	PURPOSE
`tokens/colors.css`	Raw and semantic colors, plus light-surface mapping
`tokens/typography.css`	Families, available weights, tracking, leading, and scale
`tokens/spacing.css`	Spacing, radii, strokes, and motion
`tokens/base.css`	Focus, technical text, and reduced motion
`components/components.css`	Cards, buttons, fields, badges, and schedule rows

ACCESSIBILITY

- Status includes text or shape.
- Interactive controls receive a visible 2 px focus ring.
- Light surfaces use deeper accent values for text.
- Motion respects the operating-system preference.

RELATIONSHIP AND IMPLEMENTATION

A ShruggieTech project

go-schedule shares the parent brand's typography, dark-first discipline, and direct technical writing. Its mint-and-blue scheduling palette and terminal-field mark remain product-specific.

A SHRUGGIETECH PROJECT

Use the endorsement in footers, About views, title-page colophons, repository metadata, and social previews. Keep it visually subordinate and outside the logo clear space.

LOAD THE SYSTEM

```
<link rel="stylesheet" href="styles.css">
<link rel="stylesheet" href="components/components.css">
```

The default surface is dark. Add `class="gs-light"` to a container for the accessible light token mapping.

BROWSER

- SVG favicon
- Seven-entry ICO
- 16 to 512 px PNGs
- Apple touch icon
- Web manifest

PLATFORMS

- Windows ICO
- macOS ICNS
- Linux hicolor tree
- Linux desktop-entry template
- Repository header and social preview

SOURCE OF TRUTH

The generation scripts, bundled fonts, and `build/geometry.py` recreate every logo and platform asset. `build/verify.py` checks portability, dimensions, contrast, PDF structure, encoding, and checksums.

INVENTORY

What ships

The kit contains editable source, portable masters, platform derivatives, implementation tokens, examples, and measured verification.

DIRECTORY OR FILE	CONTENTS
`logos/svg/`	Vector masters, lockups, marks, wordmarks, repository header, social preview
`logos/png/`	High-resolution raster exports
`favicons/`	Browser, Apple, Android, SVG, ICO, and manifest assets
`platform/`	Windows, macOS, and Linux deliverables
`fonts/`	WOFF2, TTF, CSS, and OFL licenses
`tokens/`	JSON source and CSS implementation tokens
`icons/`	Six scheduling icons
`components/`	Framework-neutral CSS and React wrappers
`guidelines/`	Live visual reference
`ui_kits/`	Demonstration task dashboard
`specimens/`	Outlined schedule-readability specimen
`build/`	Reproducible generators and verification
`SKILL.md`	Compact agent and contributor instructions
`brand-guide.pdf`	Printable reference manual
`VERIFY.md`	Measured checks and SHA-256 inventory

PREFERRED FORMAT

Use SVG in interfaces and documentation. Use PNG for social platforms, raster-only systems, and external listings.

VERIFICATION

Run the commands in `build/README.md`. A nonzero verification result means the kit has drifted from one of its own claims.