SCHEDULING CRON JOBS IN OPENCLAW

The Complete Walkthrough — Gateway Scheduler, CLI & Beyond

ScaleUP Media | SPRINT Training | February 2026

WHAT CAN YOU AUTOMATE?

OpenClaw's Gateway has a built-in cron scheduler. Jobs persist, survive restarts, and deliver output to any channel.

Morning Briefings

Weather, calendar, emails — delivered at 7 AM

Inbox & Data Monitors

Scan for urgent items on a schedule

Health Checks

Model/system status every 5 minutes

Recurring Reports

Weekly summaries via Slack, WhatsApp, Telegram

One-Shot Reminders

Auto-delete after run with --at "20m"

Background Chores

Isolated sessions — no main chat clutter

Jobs stored at ~/.openclaw/cron/jobs.json — persists across restarts

OPENCLAW CRON — KEY CONCEPTS

01

Three Schedule Types

"at" (one-shot timestamp) • "every" (interval in ms) • "cron" (5-field expression + timezone)

02

Main vs Isolated Sessions

Main: runs in your heartbeat context. Isolated: fresh session in cron:<jobId> — no chat clutter.

03

Delivery Modes

"announce" delivers output to WhatsApp/Telegram/Slack/Discord. "none" runs silently.

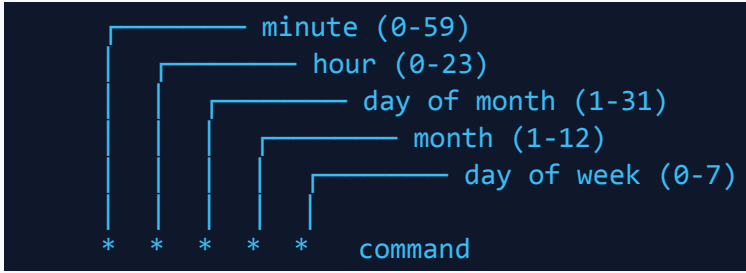
04

Model & Thinking Overrides

Use --model opus --thinking high per job. Cheaper models for simple tasks, Opus for deep analysis.

Cron runs INSIDE the Gateway — not inside the model. The Gateway persists jobs and wakes the agent.

CRON SYNTAX — THE FOUNDATION



OPERATORS

```
*   Any/every value
,   Value list (1,15)
-   Range (1-5 = Mon-Fri)
/   Step (*/5 = every 5th)
```

COMMON SCHEDULES

Expression	When It Runs
`* * * * *`	Every minute
`*/5 * * * *`	Every 5 minutes
`0 * * * *`	Top of every hour
`0 0 * * *`	Daily at midnight
`0 7 * * *`	Daily at 7 AM
`0 8 * * 1-5`	Weekdays at 8 AM
`0 9 * * 1`	Every Monday 9 AM

OPENCLAW SCHEDULE KINDS

Kind	Usage
`at`	One-shot: --at "2026-02-01T16:00:00Z"
`at (relative)`	Relative: --at "20m" or --at "2h"
`every`	Interval: --every "30m" (ms internally)
`cron`	5-field: --cron "0 7 * * *" --tz ...
`(timezone)`	Uses croner; defaults to host TZ

PRO TIP: OpenClaw uses "croner" for parsing. If timezone is omitted, the Gateway host's local TZ is used.

METHOD 1 — THE NATIVE WAY

OPENCLAW GATEWAY CRON (CLI)

Built into the Gateway. Persists at ~/.openclaw/cron/. Survives restarts. Delivers to any channel.

ONE-SHOT REMINDER (auto-deletes after run)

```
openclaw cron add \
  --name "Reminder" \
  --at "2026-02-01T16:00:00Z" \
  --session main \
  --system-event "Reminder: check the cron docs draft" \
  --wake now --delete-after-run
```

RECURRING ISOLATED JOB (delivers to Slack)

```
openclaw cron add \
  --name "Morning brief" \
  --cron "0 7 * * *" \
  --tz "America/Los_Angeles" \
  --session isolated \
  --message "Summarize overnight updates." \
  --announce \
  --channel slack --to "channel:CHANNELID"
```

MANAGEMENT

```
openclaw cron list                    # List all jobs
openclaw cron run <jobId>              # Force-run immediately
openclaw cron edit <jobId> --model opus --thinking high
openclaw cron runs --id <jobId> --limit 50    # Run history
```

MAIN SESSION vs ISOLATED SESSION

MAIN SESSION

`payload.kind = "systemEvent"`

- Enqueues a system event into heartbeat
- Full main-session context available
- wakeMode: "now" = immediate heartbeat
- wakeMode: "next-heartbeat" = waits
- Best for: reminders, context-aware tasks

```
openclaw cron add \
  --name "Calendar check" \
  --at "20m" \
  --session main \
  --system-event "Check calendar." \
  --wake now
```

ISOLATED SESSION

`payload.kind = "agentTurn"`

- Runs in fresh session cron:<jobId>
- No conversation carry-over
- Default delivery: "announce" (summary)
- Can override model + thinking level
- Best for: briefings, reports, background chores

```
openclaw cron add \
  --name "Deep analysis" \
  --cron "0 6 * * 1" \
  --session isolated \
  --message "Weekly analysis." \
  --model opus --thinking high
```

CRON vs HEARTBEAT: Use heartbeats for batched periodic checks (inbox + calendar + weather in one pass).
Use cron for exact-time tasks that need isolation, model overrides, or channel delivery.

DELIVERY — WHERE OUTPUT GOES

Isolated jobs can deliver directly to any connected channel. No heartbeat required for delivery.

WhatsApp

```
--to "+15551234567"
```

Telegram

```
--to "-1001234567890:topic:123"
```

Slack

```
--to "channel:C1234567890"
```

Discord

```
--to "channel:123456789"
```

Signal

```
--to "+15551234567"
```

iMessage

```
--to "+15551234567"
```

FULL EXAMPLE: MORNING BRIEFING → WHATSAPP

```
openclaw cron add \
  --name "Morning briefing" \
  --cron "0 7 * * *" --tz "America/New_York" \
  --session isolated \
  --message "Generate today's briefing: weather, calendar, ..." \
  --model opus \
  --announce \
  --channel whatsapp --to "+15551234567"
```

GATEWAY TOOL CALLS (JSON API)

The agent can also create cron jobs via tool calls — these are the JSON shapes the Gateway accepts.

cron.add — One-Shot (Main)

```
{
  "name": "Reminder",
  "schedule": {
    "kind": "at",
    "at": "2026-02-01T16:00:00Z"
  },
  "sessionTarget": "main",
  "wakeMode": "now",
  "payload": {
    "kind": "systemEvent",
    "text": "Reminder text"
  },
  "deleteAfterRun": true
}
```

cron.add — Recurring (Isolated)

```
{
  "name": "Morning brief",
  "schedule": {
    "kind": "cron",
    "expr": "0 7 * * *",
    "tz": "America/Los_Angeles"
  },
  "sessionTarget": "isolated",
  "payload": {
    "kind": "agentTurn",
    "message": "Summarize updates."
  },
  "delivery": {
    "mode": "announce",
    "channel": "slack"
  }
}
```

GATEWAY API: `cron.list` • `cron.add` • `cron.update` • `cron.remove` • `cron.run` • `cron.runs` • `cron.status`

PRODUCTION RECIPES

Morning Briefing
`0 7 * * *`

Weather + calendar + emails → WhatsApp

Email Monitor
`*/15 * * * *`

Scan inbox for urgent items → alert

Weekly Review
`0 9 * * 1`

Deep project analysis with --model opus

Meeting Reminder
`--at "20m"`

One-shot, main session, auto-deletes

Nightly Summary
`0 22 * * *`

Day recap → Telegram topic thread

GHL Content Gen
`0 10 * * 1,3,5`

AI social posts → webhook delivery

Project Health
`--every "4h"`

System event + wake now for check-in

Codebase Analysis
`0 6 * * 0`

Isolated + opus + thinking high

CRON vs HEARTBEAT — WHEN TO USE EACH

	Heartbeat	Cron Job
Session	Main (shared context)	Main or Isolated
Timing	Regular interval (30m default)	Exact time / cron expression
Context	Full conversation history	Fresh per run (isolated)
Batching	Multiple checks in one turn	One task per job
Model	Default agent model	Override per job
Delivery	Via heartbeat response	Direct to channel
Best for	Reactive monitoring	Proactive, time-specific tasks

RULE OF THUMB: Heartbeats = "check if anything needs attention"
Cron jobs = "do this specific thing at this specific time"

TROUBLESHOOTING & BEST PRACTICES

COMMON ISSUES

"Nothing runs"

```
Check cron.enabled in config
Ensure Gateway is running continuously
Verify timezone (--tz) vs host TZ
```

Job keeps delaying

```
Exponential backoff: 30s→1m→5m→15m→60m
Backoff resets after next success
One-shot jobs disable, don't retry
```

Wrong delivery target

```
Telegram topics: use -100…:topic:<id>
Slack/Discord: use channel:<id> prefix
Check "last route" if using --channel last
```

Main session job silent

```
Main session jobs need heartbeat enabled
Check wakeMode: "now" vs "next-heartbeat"
Verify HEARTBEAT.md exists in workspace
```

BEST PRACTICES

1. Start with 3 jobs: briefing, email monitor, weekly review — iterate before adding more
2. Use isolated sessions for noisy/frequent tasks to keep main chat clean
3. Use --model for cost control: cheap models for simple tasks, opus for deep analysis
4. Set --tz explicitly — don't rely on host timezone detection
5. Use --delete-after-run for one-shot reminders
6. Check openclaw cron list --verbose to verify next run times
7. Use heartbeats as a watchdog for your cron jobs (check if critical jobs ran)
8. Prefer CLI (openclaw cron add) over chat-created jobs for reliability
9. Use bestEffort: true on delivery to avoid job failure on channel issues
10. Keep delivery.to explicit — avoid relying on "last route" for production jobs

QUICK REFERENCE

Command	What It Does
`openclaw cron add --name ... --cron ...`	Create a new cron job
`openclaw cron list`	List all jobs with status
`openclaw cron run <jobId>`	Force-run a job immediately
`openclaw cron edit <jobId> --message ...`	Update an existing job
`openclaw cron runs --id <jobId>`	View run history
`openclaw cron rm <jobId>`	Delete a job
`openclaw system event --mode now --text ...`	Fire one-off event (no job)

CONFIGURATION

```
{ cron: {
    enabled: true,
    store: "~/.openclaw/cron/jobs.json",
    maxConcurrentRuns: 1
} }
```

DISABLE CRON

```
# Config: cron.enabled: false
# Env: OPENCLAW_SKIP_CRON=1

# Storage: ~/.openclaw/cron/
# History: ~/.openclaw/cron/runs/
```