Automation

Cron vs Heartbeat

Both heartbeats and cron jobs let you run tasks on a schedule. This guide helps you choose the right mechanism for your use case.

Quick Decision Guide

Use Case	Recommended	Why
Check inbox every 30 min	Heartbeat	Batches with other checks, context-aware
Send daily report at 9am sharp	Cron (isolated)	Exact timing needed
Monitor calendar for upcoming events	Heartbeat	Natural fit for periodic awareness
Run weekly deep analysis	Cron (isolated)	Standalone task, can use different model
Remind me in 20 minutes	Cron (main, `--at`)	One-shot with precise timing
Background project health check	Heartbeat	Piggybacks on existing cycle

Heartbeat: Periodic Awareness

Heartbeats run in the **main session** at a regular interval (default: 30 min). They're designed for the agent to check on things and surface

anything important.

When to use heartbeat

Multiple periodic checks: Instead of 5 separate cron jobs checking inbox, calendar, weather, notifications, and project status, a single heartbeat can batch all of these.

Context-aware decisions: The agent has full main-session context, so it can make smart decisions about what's urgent vs. what can wait.

Conversational continuity: Heartbeat runs share the same session, so the agent remembers recent conversations and can follow up naturally.

Low-overhead monitoring: One heartbeat replaces many small polling tasks.

Heartbeat advantages

Batches multiple checks: One agent turn can review inbox, calendar, and notifications together.

Reduces API calls: A single heartbeat is cheaper than 5 isolated cron jobs.

Context-aware: The agent knows what you've been working on and can prioritize accordingly.

Smart suppression: If nothing needs attention, the agent replies `HEARTBEAT_OK` and no message is delivered.

Natural timing: Drifts slightly based on queue load, which is fine for most monitoring.

Heartbeat example: HEARTBEAT.md checklist

```
# Heartbeat checklist

- Check email for urgent messages
- Review calendar for events in next 2 hours
- If a background task finished, summarize results
- If idle for 8+ hours, send a brief check-in
```

The agent reads this on each heartbeat and handles all items in one turn.

Configuring heartbeat

```
{
  agents: {
    defaults: {
      heartbeat: {
        every: "30m", // interval
        target: "last", // where to deliver alerts
        activeHours: { start: "08:00", end: "22:00" }, // optional
      },
    },
  },
}
```

See for full configuration.

Cron: Precise Scheduling

Cron jobs run at precise times and can run in isolated sessions without affecting main context. Recurring top-of-hour schedules are automatically spread by a deterministic per-job offset in a 0-5 minute window.

When to use cron

Exact timing required: “Send this at 9:00 AM every Monday” (not “sometime around 9”).

Standalone tasks: Tasks that don’t need conversational context.

Different model/thinking: Heavy analysis that warrants a more powerful model.

One-shot reminders: “Remind me in 20 minutes” with `--at`.

Noisy/frequent tasks: Tasks that would clutter main session history.

External triggers: Tasks that should run independently of whether the agent is otherwise active.

Cron advantages

Precise timing: 5-field or 6-field (seconds) cron expressions with timezone support.

Built-in load spreading: recurring top-of-hour schedules are staggered by up to 5 minutes by default.

Per-job control: override stagger with `--stagger <duration>` or force exact timing with `--exact`.

Session isolation: Runs in `cron:<jobId>` without polluting main history.

Model overrides: Use a cheaper or more powerful model per job.

Delivery control: Isolated jobs default to `announce` (summary); choose `none` as needed.

Immediate delivery: Announce mode posts directly without waiting for heartbeat.

No agent context needed: Runs even if main session is idle or compacted.

One-shot support: `--at` for precise future timestamps.

Cron example: Daily morning briefing

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

This runs at exactly 7:00 AM New York time, uses Opus for quality, and announces a summary directly to WhatsApp.

Cron example: One-shot reminder

```
openclaw cron add \
  --name "Meeting reminder" \
  --at "20m" \
  --session main \
  --system-event "Reminder: standup meeting starts in 10 minutes." \
  --wake now \
  --delete-after-run
```

See for full CLI reference.

Decision Flowchart

```
Does the task need to run at an EXACT time?
  YES -> Use cron
  NO  -> Continue...
            ›

Does the task need isolation from main session?
  YES -> Use cron (isolated)
  NO  -> Continue...

Can this task be batched with other periodic checks?
  YES -> Use heartbeat (add to HEARTBEAT.md)
  NO  -> Use cron

Is this a one-shot reminder?
  YES -> Use cron with --at
  NO  -> Continue...

Does it need a different model or thinking level?
  YES -> Use cron (isolated) with --model/--thinking
  NO  -> Use heartbeat
```

Combining Both

The most efficient setup uses **both**:

1. **Heartbeat** handles routine monitoring (inbox, calendar, notifications) in one batched turn every 30 minutes.
2. **Cron** handles precise schedules (daily reports, weekly reviews) and one-shot reminders.

Example: Efficient automation setup

HEARTBEAT.md (checked every 30 min):

```
# Heartbeat checklist

- Scan inbox for urgent emails
- Check calendar for events in next 2h
- Review any pending tasks
- Light check-in if quiet for 8+ hours
```

Cron jobs (precise timing):

```
# Daily morning briefing at 7am
openclaw cron add --name "Morning brief" --cron "0 7 * * *" --session isolated --

# Weekly project review on Mondays at 9am
openclaw cron add --name "Weekly review" --cron "0 9 * * 1" --session isolated --

# One-shot reminder
openclaw cron add --name "Call back" --at "2h" --session main --system-event "Cal
```

Lobster: Deterministic workflows with approvals

Lobster is the workflow runtime for **multi-step tool pipelines** that need deterministic execution and explicit approvals. Use it when the task is more than a single agent turn, and you want a resumable workflow with human checkpoints.

When Lobster fits

Multi-step automation: You need a fixed pipeline of tool calls, not a one-off prompt.

Approval gates: Side effects should pause until you approve, then resume.

Resumable runs: Continue a paused workflow without re-running earlier steps.

How it pairs with heartbeat and cron

Heartbeat/cron decide *when* a run happens.

Lobster defines *what steps* happen once the run starts.

For scheduled workflows, use cron or heartbeat to trigger an agent turn that calls Lobster. For ad-hoc workflows, call Lobster directly.

Operational notes (from the code)

- Lobster runs as a **local subprocess** (`lobster` CLI) in tool mode and returns a **JSON envelope**.
- If the tool returns `needs_approval` , you resume with a `resumeToken` and `approve` flag.
- The tool is an **optional plugin**; enable it additively via `tools.alsoAllow: ["lobster"]` (recommended).
- If you pass `lobsterPath` , it must be an **absolute path**.

See **Lobster** for full usage and examples.

Main Session vs Isolated Session

Both heartbeat and cron can interact with the main session, but differently:

	Heartbeat	Cron (main)	Cron (isolated`
Session	Main	Main (via system event)	`cron:<jobId>`
History	Shared	Shared	Fresh each run
Context	Full	Full	None (starts clean)
Model	Main session model	Main session model	Can override
Output	Delivered if not `HEARTBEAT_OK`	Heartbeat prompt + event	Announce summary (default)

When to use main session cron

Use `--session main` with `--system-event` when you want:

- The reminder/event to appear in main session context
- The agent to handle it during the next heartbeat with full context
- No separate isolated run

```
openclaw cron add \
  --name "Check project" \
  --every "4h" \
  --session main \
  --system-event "Time for a project health check" \
  --wake now
```

When to use isolated cron

Use `--session isolated` when you want:

- A clean slate without prior context
- Different model or thinking settings
- Announce summaries directly to a channel

History that doesn't clutter main session

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

Cost Considerations

Mechanism	Cost Profile
Heartbeat	One turn every N minutes; scales with HEARTBEAT.md size
Cron (main)	Adds event to next heartbeat (no isolated turn)
Cron (isolated)	Full agent turn per job; can use cheaper model

Tips:

- Keep `HEARTBEAT.md` small to minimize token overhead.
- Batch similar checks into heartbeat instead of multiple cron jobs.
- Use `target: "none"` on heartbeat if you only want internal processing.
- Use isolated cron with a cheaper model for routine tasks.

Related

- full heartbeat configuration
- full cron CLI and API reference
- system events + heartbeat controls

‹ Cron Jobs Automation Troubleshooting ›