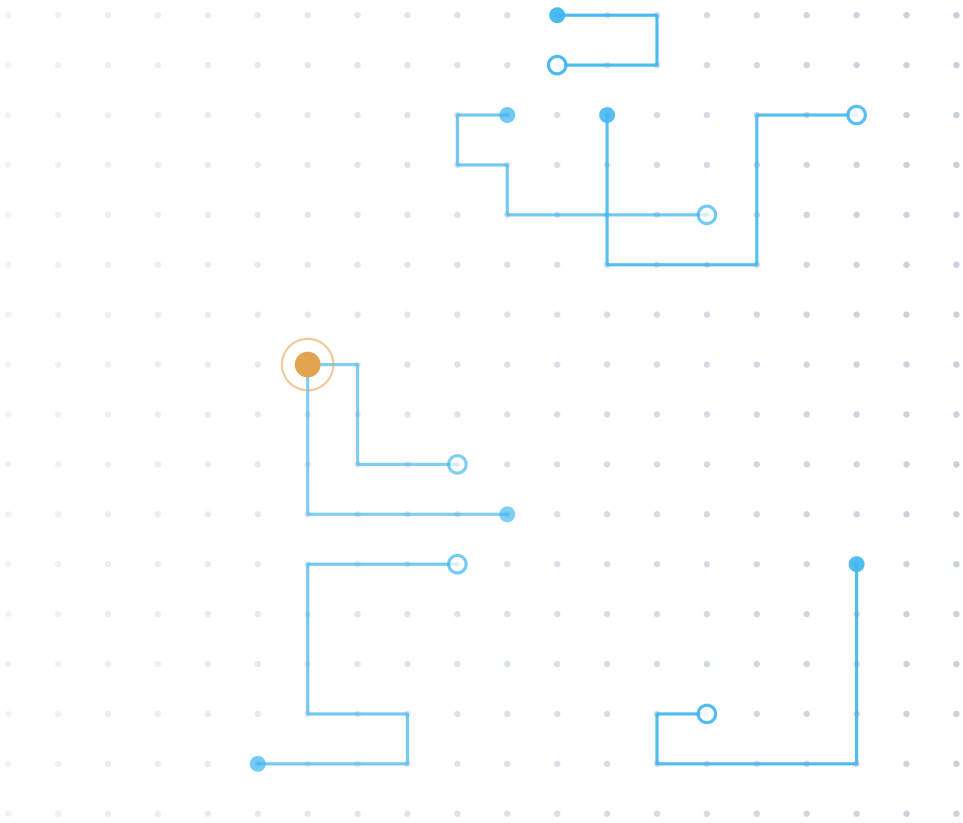


PRACTICAL GUIDE

Work while you sleep: AI agents on a 24/7 server

Let the night prepare drafts, backups and tests, and wake up to a report that tells you what happened

AUTHOR
Mario Chasco · Flowara AI

FOR
business owners and technical freelancers

READING
8 min

EDITION
v2.0 · November 2026

BEFORE YOU START

What you will get from this guide

01 Tell apart the work that can run overnight from the work that waits for a person.

02 Launch night jobs with limits that cannot take the server down.

03 Chain jobs so the machine never sits idle.

04 Read a morning report that says what was done and what was not.

CONTENTS

01 Eight idle hours every night

02 What to leave running overnight, and what never

03 Only schedule what already works by hand

04 The night rules

05 Chaining jobs: no empty promises

06 The morning report

07 Example: a one-person agency

08 Checklist

For: business owners and technical freelancers · 8 min read

01

Eight idle hours every night

A baker does not start from scratch at five in the morning. The dough proves overnight, so half the work is done by the time they arrive. Nobody stays up watching it. They trust a recipe that works. A server that is on 24 hours a day can do the same with your work.
A *24/7 server* is a computer rented in a data centre that never switches off, so it is still running at three in the morning when your laptop is closed. An *AI agent* is a program that uses a language model to plan steps and carry them out, such as writing drafts or running tests. Together they keep working while you sleep.

I run a server like this. It has 16 GB of memory and several working sessions share it at once. At night it gets on with video renders and long batch jobs, and in the morning I read what happened. The rules in this guide are the ones I have needed, and I learned several of them from a failure.

Most people end up at one of two extremes: either the machine sits idle all night, or someone leaves a job running with no rules and finds the server frozen in the morning. In the worst case, an email has gone out to a customer at three in the morning and nobody knows why.

IN ONE LINE

This guide is the middle ground: night work with rules, and a clear report to read over coffee.

02

What to leave running overnight, and what never

The question is not "can the AI do it?" but "what happens if it gets it wrong and nobody sees until nine?". If the answer is "nothing serious, I will check it in the morning", it is night work. If the answer involves a customer or money, or something that cannot be undone, it waits until someone is awake.
Take a hair salon. Preparing tomorrow's appointment reminders overnight is fine if they stay as drafts. Sending them at three in the morning is not.

Two questions are enough to place any task:

FIGURE 1 **What can run overnight**

WHERE IT ENDS UP ↓ / IF IT GOES WRONG →	CAN BE UNDONE	NO WAY BACK
STAYS ON THE MACHINE	**Overnight, no worries** drafts, reports, tests and renders	**Only with someone awake** deleting data or changing permissions
GOES OUT	**Waits for approval** publishing changes to the website	**Never overnight** messages to customers and payments

The underlying rule fits on one line:

> *At night the AI prepares; during the day a person decides.*
>
> FLOWARA AI WORKING RULE

Anything that goes out waits for someone awake to approve it. A badly written draft takes a minute to fix. A message sent to a customer cannot be taken back.

IN ONE LINE

At night you produce and check; anything that goes out or has no way back waits for the morning.

03

Only schedule what already works by hand

A common mistake is scheduling for the night something nobody has seen work during the day. If it fails by hand, it will fail at night too, only with nobody watching. Think of a summary of the day's emails. First you ask for it yourself every afternoon for a few days and check that it gets things right. Only then do you schedule it. The path is short and always goes in the same order:

FIGURE 2 **From a manual task to a night job**

1. **Do it by hand**
   One person runs it and checks every result.
2. **Repeat it unchanged**
   Several times in a row, without tweaking anything in between.
3. **Write the recipe**
   A script with the steps and what it must leave done.
4. **Schedule it**
   By time, at the same hour each night, or by event, when a file arrives.
5. **Give it a cap**
   A daily spending limit per client; when reached, it stops and alerts.

The spending cap matters more than it seems. An agent stuck in a loop in the small hours can spend a lot on model calls before anyone notices. With a daily cap per client, it does not get far. It stops, and the case goes to a person.

Two more rules come from security. Every agent and every person logs in to the server with their own identity, never a shared account. If something happens, you know who did it and can cut off that one access without touching anyone else. And if the night job touches a live website, it changes nothing directly. It proposes the change for review (a *pull request*) and, once a person approves it, deployment is automatic.

Finally, every agent has a named person who answers for it. If nobody is responsible, nobody reads its report.

IN ONE LINE

If it does not work by hand during the day, it does not get scheduled for the night.

04 The night rules

A shared server is like a flat with housemates. Run the washing machine and the oven at the same time at night and the fuse trips. Everything goes off, including whatever someone else was doing. That is why every night job follows five rules:

RULE	WHAT IT ACHIEVES	HOW YOU SET IT
Low priority	Gives way to anything else that needs the processor	`Nice=19`
Capped CPU and memory	If it wants more, it waits instead of taking over	`CPUQuota` and `MemoryMax`
One heavy job at a time	Never two renders together	A queue, one after another
Resumable	If it stops, it carries on from the last item	Progress saved item by item
Dies first	If memory runs short, it goes before anything that matters	`OOMScoreAdjust=500`

On Linux, most of that fits in a single command that puts the job in a box with limits. Before launching, check free memory; if it is tight, use a smaller batch:

BASH

```
# Before launching: how much memory is free?
free -h
# Night job: lowest priority, capped, first to go if memory runs out
systemd-run --user --collect \
  -p Nice=19 -p CPUQuota=50% -p MemoryMax=1G \
  -p OOMScoreAdjust=500 ./prepare-drafts.sh
```

I learned this rule the hard way. In September, a large data job launched with no cap ran my server out of memory. The system started killing processes to save itself and took the editor down with them. I lost that session's work. Since then, anything that runs for more than a few minutes runs outside the editor and with these limits; if the editor closes, the job carries on.

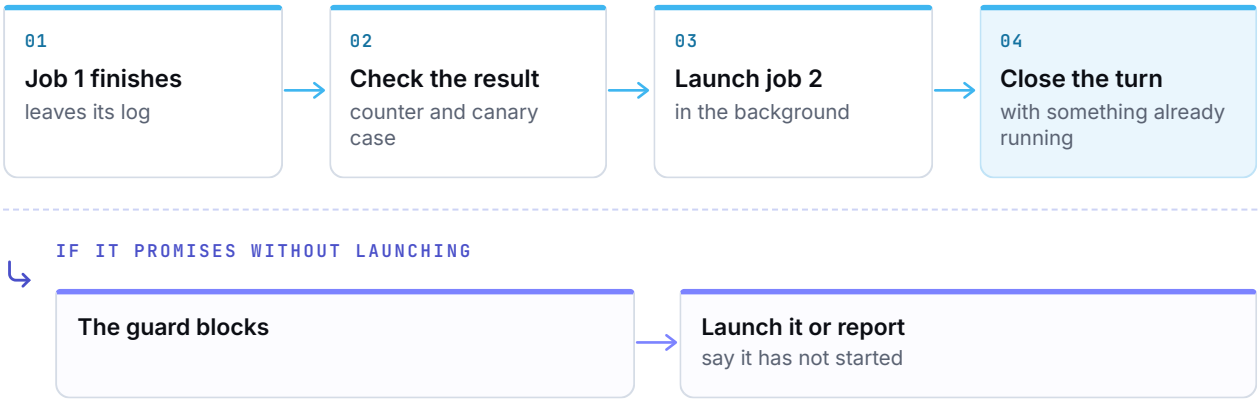
IN ONE LINE

A well-built night job cannot take the server down, not even when it fails.

05 Chaining jobs: no empty promises

The most expensive failure at night is not an error. It is silence. It happened to me. A working session ended its reply with "now I am starting the edit", launched nothing, and the server sat idle for three hours. Everything looked fine, but nothing was being done. No error log would have shown it, because technically nothing had failed.
The fix is a chain with a guard:

FIGURE 3 **A chain with a guard**



A *turn* is each cycle in which the agent replies and stops. The guard is an automatic check that reads the end of that turn, and if the turn announces work ("starting now", "proceeding") with nothing launched, it does not let the turn end. There are only two valid ways out: launch the job in the background, or say clearly that it has not started and why. My server has worked this way since that day.

> ✦ RULE
>
> When a long job finishes, the next one is launched in the same turn. Nobody should have to ask "how is it going?" for the queue to move.

IN ONE LINE

A server never sits idle after promising work.

06

The morning report

If you have to open ten folders in the morning to find out what happened, the night was wasted. Everything ends in a short report that takes two minutes to read. The numbers come first and the pending work comes last. There are five sections:

01 **Counters.** How many items there were and how many were processed, batch by batch.

02 **Done** Green. What is ready for review, with a direct link to each result.

03 **Failed** Red. The step that failed and whether it was retried.

04 **Skipped** Amber. What was left out on purpose, with the reason (not enough memory, or a missing input).

05 **What I did not do.** What was in the plan and was not touched. It is the most honest section and the one most often forgotten.

No failure can be silent. Every job leaves a log that someone reads and a counter that should move. It also carries a *canary case*: an example whose result you know in advance. If the canary comes out wrong, do not trust the rest.

> ⚠ CAUTION
>
> "all fine" with no numbers is not a report. If it does not say how many, you cannot tell whether two hundred items were processed or none.

IN ONE LINE

A good report tells you what was done, what failed and what never got started.

07

Example: a one-person agency

A freelancer runs a small marketing agency on her own for shops in her town (synthetic example). Before closing her laptop, she queues three jobs on her server:

01 **Post drafts** for four clients, built from the week's notes and each brand's rules.

02 **A synced backup** of her project folders to external storage, checking that nothing is missing.

03 **Automated tests** of the contact forms on the websites she maintains.

Each job runs at low priority with a memory cap, one after another. At 8:15 she reads this:

> **EXAMPLE**
>
> Drafts: 16 of 16, ready for review. Backup: 2,340 files, 0 differences. Tests: 11 of 12 passed; the florist's form fails (configuration error). Skipped: the garage video, the script was missing. Did not do: posts for the new client, who has no brand rules yet.

In twenty minutes she approves fourteen drafts, edits two and fixes the form before any customer notices. Nothing went out without her approval. What matters is not that everything went well, but that the report told her straight away where to look.

FIGURE 4 **The morning in three numbers**

16 of 16	11 of 12	20 min
drafts ready for review	contact form tests passed	of review over coffee

Source: one morning's report (synthetic example)

IN ONE LINE

The night does the work; she makes the decisions over coffee.

08 Checklist

Before you leave the first job running overnight, go through this:

Before the first night

16 ITEMS

- [ ] Every night job prepares or checks
  none sends, charges or deletes
- [ ] Anything that goes out waits for a person who is awake
- [ ] Only what already works by hand gets scheduled
- [ ] There is a daily spending cap per client
  when reached, a person takes over
- [ ] Each agent logs in with its own identity
  never a shared account
- [ ] Changes to live sites go through review
  automatic deployment after approval
- [ ] Low priority and capped CPU and memory
- [ ] Only one heavy job at a time
- [ ] Free memory is checked before launching
  if tight, a smaller batch
- [ ] Every job saves its progress and can resume
- [ ] It runs outside the editor or session
  survives if they close
- [ ] The next job is launched in the same turn
- [ ] A guard stops a turn that promises work without launching it
- [ ] Every job leaves a log, a counter and a canary case
- [ ] The report includes what failed and what was not done
- [ ] Every agent has a named person who answers for it

NEXT STEP

Free diagnostic (20 min)

If you want to see which part of your business can be automated with the risk under control, book a free 20-minute diagnostic.

Book a diagnostic ›

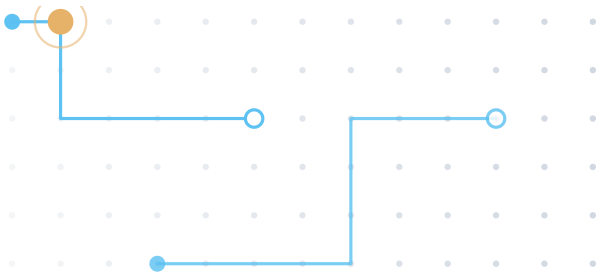
flowaraai.com/en/contact

RELATED RESOURCES

Claude Code toolkit on GitHub ↗

I Left Claude Code Working All Night: The Morning Report ↗

Five Claude Code Automations a Small Business Can Set Up This Week ↗



Mario Chasco · Flowara AI

AI systems in production for small businesses.

flowaraai.com
github.com/mariochasco78557
youtube.com/@mariochasco_AI
linkedin.com/in/mario-adrian-chasco-frauca

Share this guide as is, crediting flowaraai.com.