Thirty-four lines of shell to keep a local model alive

Strata's server belongs to its launcher, so a plain systemd unit does not work. A one-shot unit plus a small watchdog on a timer had the model answering again 32 seconds after the watchdog noticed I had killed it.



Published
1 November 2026

Reading time
4 minutes

About
Mullet

Tags
Linux, AI

More on: Mullet

Mullet's local rung only works while the local model answers. On 1 November I wanted Strata to start at boot and come back after a crash without me. It took a systemd unit, a watchdog, and one trap in testing it.

Why not a normal unit

`strata launch --preset NAME --no-tui --yes --restart` starts Strata's runtime host and server, and returns once the model answers. The server is owned by that host, so a plain unit that runs the server command line does not work. The unit has to wrap the launcher instead.

Everything here is per user and uses `systemctl --user`. Replace the paths, the preset name and the port with your own. This is `~/.config/systemd/user/strata-coder.service`, with my paths replaced and my comments left out:

```
[Unit]
Description=Strata: Qwen3.8-Flash-Next Coder (preset coder-128k) on
127.0.0.1:8080
```

```
[Service]
Type=oneshot
RemainAfterExit=yes
TimeoutStartSec=900
TimeoutStopSec=120
Nice=5
ExecStart=/path/to/strata launch --preset coder-128k --no-tui --yes --
restart --data-dir /path/to/data
ExecStop=/path/to/strata launch --stop --preset coder-128k --data-dir
/path/to/data

[Install]
WantedBy=default.target
```

The catch is that a one-shot stays “active” after the launcher returns. If the server dies, systemd does not notice. That is why there is a watchdog.

The watchdog

`~/.local/bin/strata-watchdog`, made executable. The real file is 34 lines; this is the same code with my comments left out:

```
#!/bin/sh
UNIT=strata-coder.service
URL=${STRATA_URL:-http://127.0.0.1:8080/v1/models}
STATE=${XDG_RUNTIME_DIR:-/tmp}/strata-watchdog.fails

state=$(systemctl --user is-active "$UNIT" 2>/dev/null)
[ "$state" = inactive ] && exit 0      # stopped on purpose
[ "$state" = activating ] && exit 0    # still starting

if curl -sf --max-time 15 "$URL" >/dev/null; then rm -f "$STATE"; exit 0;
fi

if pgrep -f "serve.server --engine strata" >/dev/null; then
  fails=$(( $(cat "$STATE" 2>/dev/null || echo 0) + 1 ))
  if [ "$fails" -lt 3 ]; then echo "$fails" > "$STATE"; exit 0; fi
  reason="the API did not answer for $fails checks in a row"
else
  reason="the server process is gone"
fi

rm -f "$STATE"
logger -t strata-watchdog "restarting $UNIT: $reason"
systemctl --user restart "$UNIT"
```

The rules are short. If the unit is inactive, I stopped it on purpose, so do nothing. If it is activating, it is still starting, so do nothing. If the API answers, clear the failure count. If the server process is gone, restart at once. If the process exists but the API is silent, count the failure in a file and restart on the third, so a long request on a busy one-slot server is not mistaken for a hang.

A timer runs it. `strata-coder-watchdog.service` is a one-shot whose `ExecStart` is `%h/.local/bin/strata-watchdog`, and the timer is:

```
[Timer]
OnBootSec=4min
OnUnitActiveSec=2min
Unit=strata-coder-watchdog.service

[Install]
WantedBy=timers.target
```

Then `systemctl --user daemon-reload`, `systemctl --user enable --now strata-coder.service strata-coder-watchdog.timer`, and `loginctl enable-linger "$USER"`. Linger was already on here, so the model starts at boot without a login.

The test

I killed the server and the engine with `kill -9`. The API returned nothing, and the unit still said active, which is exactly why the watchdog is needed. The watchdog logged “restarting strata-coder.service: the server process is gone”, and the model was answering again 32 seconds after the watchdog started (the journal has the watchdog starting at 19:42:29 and Strata reporting the model answering at 19:43:02 on 1 November). The model files were already in the page cache. That is one test, a sample of one.

The timer adds up to two minutes before the watchdog notices, so a crash costs up to about 2 minutes 32 seconds (my arithmetic). A hang takes up to about 6 minutes to notice: three failed checks two minutes apart, and the first may come up to two minutes after the hang starts. That is my estimate from the timings. I tested a crash, not a hang. With cold files, such as after a reboot, Strata warns that the start can take one to three minutes rather than 30 seconds or so.

A trap when testing it

`pkill -f` and `pgrep -f` match whole command lines. If a test script runs `pkill -f "serve.server --engine strata"` through a shell, that shell’s own command line contains the same text, so the pattern matches the shell as well. In my test script that killed the script halfway through, and a `pgrep -f` check reported the server as running for the same reason: it found the shell asking the question.

The fix is the bracket trick. Write `[s]erve.server` in the pattern. As a regular expression it still matches `serve.server`, but the literal text in the command line no longer does. The watchdog is safe without it, because its pattern lives in the script file, not on its command line.

The full units are in the public repo, in `docs/STRATA_SERVICE.md`, at https://github.com/tomgould/mullet-public (https://github.com/tomgould/mullet-public).