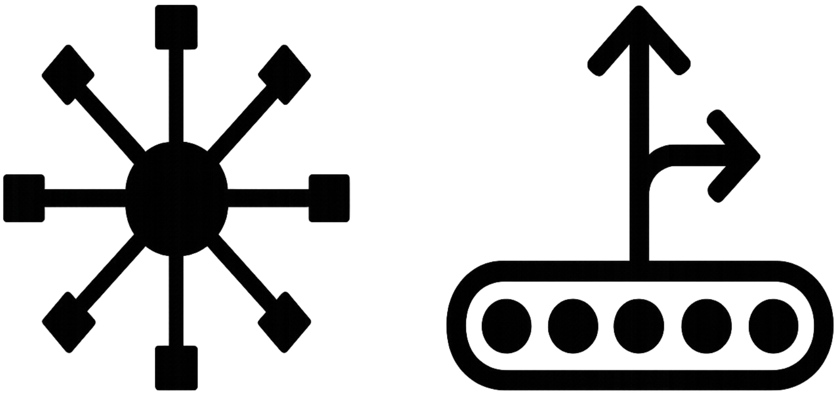
Getting the Dispatcher Right: Checklist & Blueprint!



How to “get the dispatcher right” — a step-by-step recipe

Layer	What right looks like	Key design moves & code hooks

1 · Contract (What files appear; where outputs land)	• One requests inbox (`requests/`) and one logs outbox (`logs/`).
	• Each request file is immutable once read (rename to `processed/...` or embed a UUID in the filename).
		Example constants:
		`bash`
		`INBOX="requests"`
		`OUTBOX="logs"`
		`DONE="processed"`
		`next_request() { ls -1 "$INBOX" | head -n1; }`

2 · Parser *(Turn the file into an “action” struct)*	• Support **one** canonical format (start with YAML).
	• Fail fast & log if the schema is wrong.
		`python`
		`import yaml, sys, json, jsonschema`
		`schema = {...}`
		`doc = yaml.safe_load(open(sys.argv[1]))`
		`jsonschema.validate(doc`

		`, schema)`
		`print(json.dumps(doc))`

3 · Router / Plugin Map	• A dictionary `kind → handler` that's discoverable at runtime (env var, entry-point, or folder scan).
	• Handlers are *pure functions*: `(request) -> result`.
		`bash`
		`case "$kind" in`
		`deploy)`
		`handler="./handlers/deploy.sh" ;;`
		`backup)`
		`handler="./handlers/backup.py" ;;`
		`*) echo "Unknown kind"; exit 64 ;;`
		`esac`

4 · Execution Sandbox	• Run as a non-root user.
	• Pass secrets via env vars or a mounted secrets file (never bake into repo).
	• Time-box & memory-box with `timeout`, cgroups, or Docker.
		`bash`
		`sudo -u executor timeout 900 "$handler" "$req_file" >"$tmp_log" 2>&1`
		`rc=$?`

5 · Idempotence & Locking	• Serialise **per resource**, not globally (use `flock` on resource ID).
	• Handlers should be safely re-runnable.
		`bash`
		`flock "/var/lock/$resource" "$handler" …`

6 · Observability	• **Structured log** (`JSONL`) + human tail (ansi).
	• Write a status stub even on crash (`exit trap`).
	• Bump a Prometheus counter each run.
		`bash`
		`trap 'echo "{\"ts\":\"$(date --iso)\",\"status\":\"crash\",\"req\":\"$req\"}" >> "$OUTBOX/metrics.jsonl"' ERR`

7 · Commit & Push	• Add the log file and a tiny status file (`processed/.../status.json`).
	• Push on a back-off loop; bail after N retries
		`bash`
		`git add "$log" "processed/$uuid.status"`
		`git commit -m "executor: $uuid`

	but keep state local.	`rc=$rc"`
		`git push || sleep 30`

8 · Self-health	· A watchdog systemd unit (or Kubernetes liveness probe).
	· Dispatcher exposes `--version` and `--self-test`.
		systemd example:
		`Restart=on-failure`
		`RestartSec=5`

9 · Extensibility path	· New handler = drop a file in `handlers/` + add one YAML line in schema. **· No dispatcher rewrite needed.**	Keep the router table in its own file (`handlers/index.yml`).
10 · Test harness	· “Golden” request fixtures + expected logs stored under `tests/`. · CI runs the dispatcher in Docker, mounts the repo, asserts diff==∅.	GitHub Actions example: `matrix: req=tests/*.yml`

Minimal starter dispatcher (Bash + Python hybrid)

```
#!/usr/bin/env bash
set -euo pipefail

while true; do
  git pull --ff-only origin main

  req=$(ls requests | head -n1 || true)
  [[ -z "$req" ]] && sleep 15 && continue

  uuid=$(date +%s)-$RANDOM
  kind=$(yq '.kind' "requests/$req")
  handler="handlers/$kind.sh"

  log="logs/$uuid.log"
  tmp=$(mktemp)

  sudo -u executor timeout 900 "$handler" "requests/$req" >"$tmp"
```

```
2>&1
  rc=$?

  mv "$tmp" "$log"
  mkdir -p processed && mv "requests/$req" "processed/$uuid.yml"
  echo "{\"rc\":$rc,\"finished\":\"$(date --iso)\"}" >
"processed/$uuid.status"

  git add "$log" "processed/$uuid."*
  git commit -m "executor $uuid rc=$rc"
  git push || true          # don't die on network flap
done
```

(Replace `handlers/deploy.sh`, `handlers/backup.py`, … with real scripts.)

Summary Checklist

1. **Single source of truth:** Git is your message bus *and* state DB.
2. **Explicit schema:** Fail on unknown keys; treat every request as immutable.
3. **Non-root sandbox:** Principle of least privilege + secrets only at runtime.
4. **Idempotent handlers:** You must be able to `rm -rf vm && rerun` safely.
5. **Observable & testable:** Logs, metrics, and a CI harness that replays fixtures.

Nail those five and you “get it right”. Everything else—cloud provider, language, even Git host—is interchangeable.