LARAVEL CHEAT SHEET

# Quality gates and lefthook

AI tool hooks, git hooks, CI. Only CI is a guarantee. GitLab CI, lefthook, gitleaks and Claude Code hooks for Laravel in Docker.

# Three levels

- **AI tool hook:** runs inside one agent session. Skipped by anyone not using that agent. For fast feedback while the agent edits.
- **Git hook:** runs where it is installed. Skipped with `git commit --no-verify` or by never installing it. For fixing style before the commit exists.
- **CI:** runs on every push, on the server. Nobody skips it if the branch is protected. **The guarantee.**
- A rule that matters lives in CI first. Hooks only make it cheaper to satisfy.

# CI checks, never fixes

```
-    - composer larastan || true
-    - composer pint --test || true
+    - composer larastan
+    - ./vendor/bin/pint --test
-    - npm run format || true
-    - npm run types || true
-    - npm run lint || true
+    - npm run format:check
+    - npm run types
+    - npm run lint:check
```

`composer pint` did not exist: `Command "pint" is not defined.`, exit 1, job green for 17 months.

```
"format": "prettier --write resources/",
"format:check": "prettier --check resources/",
"lint": "eslint . --fix",
"lint:check": "eslint .",
```

`--write` and `--fix` in CI fix a checkout that is thrown away. ESLint exits 0 on warnings: they stay visible, only errors block.

# One command that mirrors CI

```
# Run every quality gate in check mode, the same steps CI runs
check: setup-test-db
	docker compose exec -T app composer larastan
	docker compose exec -T app vendor/bin/pint --test
	docker compose exec -T app npm run format:check
	docker compose exec -T app npm run types
	docker compose exec -T app npm run lint:check
	docker compose exec -T app php artisan test --compact --parallel
```

Put `make check` in `CLAUDE.md` and `AGENTS.md`, so the agent runs the CI gate before it says it is done.

# lefthook.yml

```
pre-commit:
  commands:
    pint:
      glob: "*.php"
      run: docker compose exec -T app vendor/bin/pint {staged_files}
      priority: 1
      stage_fixed: true
    eslint:
      glob: "*.{js,jsx,ts,tsx}"
      run: docker compose exec -T app npx eslint --fix {staged_files}
      priority: 2
      stage_fixed: true
```

```
    prettier:
      glob: "resources/**/*.{js,jsx,ts,tsx,css}"
      run: docker compose exec -T app npx prettier --write --ignore-unknown {staged_files}
      priority: 3
      stage_fixed: true
    gitleaks:
      run: gitleaks git --pre-commit --staged --redact --verbose --no-banner
      priority: 4
```

Staged files only. Tests stay in `make check` and CI.

# Four lefthook traps

- `stage_fixed: true`, **not** `git add .` It re-stages only the files that passed `glob`, and works only in `pre-commit`.
- `docker compose exec -T`. Hooks run without a terminal. `{staged_files}` are relative to the repo root, which is also the container's working directory.
- **Order, not parallelism.** Sequential is the default; `priority` sets the order. With `parallel: true`, `eslint --fix` and `prettier --write` race on the same file.
- **gobwas globs.** `**` is one or more directories. Need zero too? `glob_matcher: doublestar`.

```
$lefthook install   # renames an existing hook to pre-commit.old
```

# gitleaks: scan history first

```
docker run --rm -v "$PWD":/repo -w /repo ghcr.io/gitleaks/gitleaks:v8.30.1 \
  git --redact --no-banner .
```

578 commits, `leaks found: 476`, all `generic-api-key`: 472 `translation_key` UUIDs, 4 fake `STRIPE_SECRET` in skill docs.

```
[extend]
useDefault = true

[[allowlists]]
targetRules = ["generic-api-key"]
regexTarget = "line"
regexes = ['''translation_key"?:\s*"?[0-9a-f]{8}-[0-9a-f]{4}-[0-9a-f]{4}-[0-9a-f]{4}-[0-9a-f]{12}''']

[[allowlists]]
targetRules = ["generic-api-key"]
paths = ['''^\.(claude|agents|cursor|github)/skills/.*\.md$''']
```

`.gitleaks.toml`: history scan says `no leaks found`, a fake AWS key is still caught (exit 1).

# gitleaks in GitLab CI

```
secrets:gitleaks:
  stage: test
  image:
    name: ghcr.io/gitleaks/gitleaks:v8.30.1
    entrypoint: [""]
  variables:
    GIT_DEPTH: 0
  cache: []
  script:
    - gitleaks git --redact --no-banner --verbose .
```

`GIT_DEPTH: 0` : new GitLab projects clone with depth 20, so gitleaks would see only 20 commits.

# Claude Code: Pint after each edit

```
"PostToolUse": [{
    "matcher": "Edit|Write",
    "hooks": [{
        "type": "command",
        "command": "\"$CLAUDE_PROJECT_DIR\"/.claude/hooks/pint-on-edit.sh"
    }]
}]
```

Inside `"hooks"` in `.claude/settings.json`.

```
file=$(jq -r '.tool_input.file_path // empty')
[[ "$file" == *.php ]] || exit 0

relative=${file#"$CLAUDE_PROJECT_DIR"/}
cd "$CLAUDE_PROJECT_DIR" || exit 0

if ! docker compose exec -T app vendor/bin/pint "$relative" > /dev/null; then
    echo "Pint could not format $relative" >&2
    exit 2
fi
```

Exit 2 on `PostToolUse` cannot undo the edit; it sends stderr back to the agent in the same turn. Does nothing for Codex, Cursor or hand-typed code.

# When not to use this

- **No CI at all:** add the CI job first. A hook without a server-side gate moves the problem to whoever forgot to install it.
- **Tests in pre-commit:** a 217-test hook per commit teaches `--no-verify`. Tests go to `make check` and CI.
- **A rule nobody will fix:** keep warnings as warnings rather than turning the first honest pipeline red.
- **One-file scripts and throwaway repos:** three levels cost more than they save.

RECAP

# The sheet in short

- → Only CI is a guarantee; hooks make it cheaper to pass
- → No `|| true`, and check-only scripts in CI
- → `make check` runs exactly what CI runs
- → lefthook: staged files, `stage_fixed`, `priority`, `exec -T`
- → Scan the whole history before gitleaks goes into CI
- → AI hooks give the agent feedback, they guarantee nothing

# Found this useful?

Save this cheat sheet for later.

Full article: dominik-dev.pl

**Share • Comment**