/sentry-iterate-pr

Source: `~/.claude/skills/sentry-iterate-pr/SKILL.md`

---

name: iterate-pr description: Iterate on a PR until CI passes. Use when you need to fix CI failures, address review feedback, or continuously push fixes until all checks are green. Automates the feedback-fix-push-wait cycle.

Iterate on PR Until CI Passes

Continuously iterate on the current branch until all CI checks pass and review feedback is addressed.

Requires: GitHub CLI (`gh`) authenticated.

Important: All scripts must be run from the repository root directory (where `.git` is located), not from the skill directory. Use the full path to the script via `${CLAUDE_SKILL_ROOT}`.

Bundled Scripts

`scripts/fetch_pr_checks.py`

Fetches CI check status and extracts failure snippets from logs.

```
uv run ${CLAUDE_SKILL_ROOT}/scripts/fetch_pr_checks.py [--pr NUMBER]
```

Returns JSON:

```
{
  "pr": {"number": 123, "branch": "feat/foo"},
  "summary": {"total": 5, "passed": 3, "failed": 2, "pending": 0},
  "checks": [
    {"name": "tests", "status": "fail", "log_snippet": "...", "run_id": 123},
    {"name": "lint", "status": "pass"}
  ]
}
```

`scripts/fetch_pr_feedback.py`

Fetches and categorizes PR review feedback using the LOGAF scale.

```
uv run ${CLAUDE_SKILL_ROOT}/scripts/fetch_pr_feedback.py [--pr NUMBER]
```

Returns JSON with feedback categorized as:

- `high` - Must address before merge (`h:`, blocker, changes requested)
- `medium` - Should address (`m:`, standard feedback)
- `low` - Optional (`l:`, nit, style, suggestion)
- `bot` - Automated comments (Codecov, Sentry, etc.)
- `resolved` - Already resolved threads

Workflow

1. Identify PR

```
gh pr view --json number,url,headRefName
```

Stop if no PR exists for the current branch.

2. Check CI Status

Run `${CLAUDE_SKILL_ROOT}/scripts/fetch_pr_checks.py` to get structured failure data.

Wait if pending: If bot-related checks (sentry, codecov, cursor, bugbot, seer) are still running, wait before proceeding—they may post additional feedback.

3. Fix CI Failures

For each failure in the script output:

1. Read the `log_snippet` to understand the failure
2. Read the relevant code before making changes
3. Fix the issue with minimal, targeted changes

Do NOT assume what failed based on check name alone—always read the logs.

4. Gather Review Feedback

Run `${CLAUDE_SKILL_ROOT}/scripts/fetch_pr_feedback.py` to get categorized feedback.

5. Handle Feedback by LOGAF Priority

Auto-fix (no prompt):

- `high` - must address (blockers, security, changes requested)
- `medium` - should address (standard feedback)

Prompt user for selection:

- `low` - present numbered list and ask which to address:

```
Found 3 low-priority suggestions:
1. [l] "Consider renaming this variable" - @reviewer in api.py:42
2. [nit] "Could use a list comprehension" - @reviewer in utils.py:18
3. [style] "Add a docstring" - @reviewer in models.py:55

Which would you like to address? (e.g., "1,3" or "all" or "none")
```

Skip silently:

- `resolved` threads
- `bot` comments (informational only)

6. Commit and Push

```
git add <files>
git commit -m "fix: <descriptive message>"
git push
```

7. Wait for CI

```
gh pr checks --watch --interval 30
```

8. Repeat

Return to step 2 if CI failed or new feedback appeared.

Exit Conditions

Success: All checks pass, no unaddressed high/medium feedback, user has decided on low-priority items.

Ask for help: Same failure after 3 attempts, feedback needs clarification, infrastructure issues.

Stop: No PR exists, branch needs rebase.

Fallback

If scripts fail, use `gh` CLI directly:

- `gh pr checks --json name,state,bucket,link`
- `gh run view <run-id> --log-failed`
- `gh api repos/{owner}/{repo}/pulls/{number}/comments`

Revision #5
Created 2026-02-18 08:40:00 UTC by John
Updated 2026-06-21 20:01:06 UTC by John