GitHub Event Routine: A PR Evaluation Tool

Most of what you really want AI to do—reviewing code, classifying problems, checking documentation—is reactive, not periodic.

Why is value found in GitHub events?

Scheduled routines are a great start, but they have one major limitation: They only run on time, not on demand. Most of what you really want AI to do—reviewing code, categorizing problems, checking documentation—is reactive, not routine.

That's where GitHub event triggers come into play. You set up your routine to run when a specific event occurs: a PR is opened, an issue is labeled, a commit is added to a specific path. The routine wakes up, performs its task, and then returns to Sleep until the next event occurs.

In this lesson, we will build one of the most common routines: the PR reviewer tool.

Trigger structure

When you select **a GitHub event** as the trigger, you will see:

1. **Repository** - the storage location where you want your routine to track your progress.
2. **Event category** - Pull request, Issue, Push, Workflow run, Release, etc.
3. **Action** - specific actions within that category (opened, synchronized, closed, labeled, etc.)
4. **Filters** - base branch, label, path, author

The filtering part is where you should spend the most time. That's the difference between a useful routine and a resource-wasting disaster.

The pitfalls of bypassing filters

This is what happens if you skip the filter section:

1. You set the trigger to "Pull request ? opened"
2. You save the routine.
3. Your repository receives a pull request from dependabot, a chore: a PR from a teammate, a draft PR from the designer, an undo PR from the release, and two document-only PRs in the next hour.

4. Your routine has been executed 6 times.
5. You are using the Pro version, so your daily limit is 5.
6. You used up your limit before lunchtime.
7. The actual PR you want considered at 2 PM is still there.

This was the most common mistake in the first week, cited directly from Anthropic documentation and community reports. And it was entirely preventable.

Filtration principles

For your PR evaluation routine, set up the following filters:

Filter	Proposed value	Reason
Base branch	`main`(or `develop`)	Only consider PRs targeting the production environment. Ignore merging the feature branch and the testing branch.
Label	`needs-review`(register)	It only starts when someone flags it. Process: User flags; Claude reviews.
Paths	`src/**/*.ts`,`src/**/*.py`	Ignore PRs that only contain documentation, only configuration, and only lock files. Match your code, not YAML.
Author	Exclude `dependabot[bot]`, `renovate[bot]`	PR bots are reviewed by humans or go through a separate, more cost-effective process.

Among these, adding a base branch + label removes 80% of the noise with just two clicks. If you have nothing else to do, just do those two things.

Prompt PR review

This is a production-level PR review prompt. It's longer than the backlog classification from Lesson 2 because it considers PR review a larger task – but it still follows the four rules from Lesson 2 (scope queries, fixed formats, hard limits, handling empty state).

```
B?n là m?t k? s? c?p cao ?ang th?c hi?n ?ánh giá code k? l??ng trên m?
t pull request. B?n s? nh?n ???c s? khác bi?t c?a PR, tiêu ??, mô t?
 và các v?n ?? ???c liên k?t làm ??u vào. Nhi?m v? c?a b?n là ??ng m?
t bình lu?n ?ánh giá duy nh?t v?i nh?ng phát hi?n c?a b?n. ?u tiên ?
ánh giá (theo th? t?): 1. Tính chính xác - code có th?c hi?n ?úng nh?
ng gì mô t? PR nói không? 2. B?o m?t - b? qua xác th?c, t?
n công SQL injection, XSS, l? bí m?t, gi?i mã không an toàn 3. Hi?u su?
t - truy v?n N+1, vòng l?p không gi?i h?n, các v?n ?? hot-path rõ ràng 4.
?? ph? test - các ???ng d?n ?ã thay ??i ?ã ???c test ch?a? Có nhánh nào ch?
a ???c ki?m tra không? 5. Phong cách - ch? báo cáo các v?n ?? v?
 phong cách gây c?n tr? vi?c hi?u, không bao gi? soi mói chi ti?t ??nh d?
ng ??u ra ?ánh giá (s? d?ng chính xác template này): ## Tóm t?t ?
```

```
ánh giá **K?t lu?n:** {ch?p thu?n | yêu c?u thay ??i | nh?
n xét} ### PR này làm gì {1-2 câu} ### Phát hi?n {??i v?i m?i phát hi?
n, hãy s? d?ng:} - **[severity]** `{file}:{line}` — {v?n ?? m?t dòng} - T?
i sao nó quan tr?ng: {m?t câu} - ?? xu?t s?a l?i: {m?t câu} M?c ??
 nghiêm tr?ng: nghiêm tr?ng | ?áng lo ng?i | soi mói ### Test {M?t câu v?
 vi?c các ???ng d?n ?ã thay ??i có ?? bao ph? ??y ?? hay không.} --- Quy t?
c: - Không bao gi? vi?t quá 10 phát hi?n. N?u có h?n 10, hãy ch?
n 10 phát hi?n hàng ??u theo m?c ?? nghiêm tr?ng và ghi chú r?ng còn có nh?
ng phát hi?n khác. - Không bao gi? ??ng các ?ánh giá chung chung "LGTM". Ch
? ?ánh giá n?u có phát hi?n. - N?u PR quá l?n ?? xem xét m?t cách có ý ngh?
a (>500 dòng thay ??i), hãy nói rõ ?i?u ?ó và yêu c?u chia nh?
. - Không bao gi? yêu c?u thay ??i ch? v? ki?u dáng. - Không bao gi? ??
y lên b?t k? nhánh nào. Ch? ??ng bình lu?n ?ánh giá.
```

A few things to note:

1. Clear priorities – prompting Claude to know what to search for, in what order.
2. Fixed rating format - the same template every time.
3. Hard limit on the number of detections (maximum 10)
4. The "no pushing" rule is clear – although the platform enforces this rule, repeating it in the prompt would prevent Claude from attempting it.
5. Clear size limits - instruct Claude to stop if the PR is too large.

The "one session per PR" model

This is a detail from the documentation that you can easily miss. When a PR goes through its lifecycle—being opened, new commits being pushed, comments being added, CI results arriving—all are separate GitHub events. If you configure each event as a separate routine run, you'll exceed your limit before lunchtime with a busy PR.

Instead, the Anthropic document describes a model as follows:

Claude opens a session for each PR, continuously updating that session (comments, CI results) to monitor progress.

One session. Multiple updates within the same session. Claude remembers the PR context between events. Your limit only counts one routine run, not 10.

To achieve this behavior, configure the routine with the "same session between events" option (the exact UI label may vary - find this option in the GitHub trigger's advanced options). Without this option, each push will trigger a new review without remembering the previous response.

Limit branching

One rule you cannot override: Routines cannot push to protected branches.

This applies to main, master, any branches you've marked as protected on GitHub, and any branches that require reviewers. The routine could be:

1. Post a comment on the pull request (PR).
2. Post a review/rating (approve/request changes)
3. Push to the feature branch
4. Open a new PR
5. Label, specify, close the issue.

Routine cannot:

1. Push directly to the main branch
2. Skip the mandatory assessment tool.
3. Press down on the protected branches.
4. Delete branches that are not its own.

This isn't a bug. This is a basic safety measure. If you want Claude to propose changes to the main branch, it has to go through a human-approved PR. It's a feature, not a restriction.

Plug-and-Play Configuration: Step-by-Step

Specific setup checklist:

1. **Routines ? New routine**
2. **Trigger** : GitHub Event
3. **Repository** : Select your repository (you will need to authorize the GitHub connector once).
4. **Event category** : pull request
5. **Action** : Opened + Synchronized (Synchronized = new commits pushed)
6. **Base branch filter:** `main`
7. **Label filter:** `needs-review` (optional, but recommended)
8. **Paths** : your code path (e.g., `src/**`, `lib/**`) - ignore `docs/**`, `*.md`, lockfile
9. **Author exclude:** `dependabot[bot]` ,`renovate[bot]`
10. **Connectors** : GitHub only
11. **Session behavior** : the same session for each PR
12. **Prompt** : prompt above
13. **Save** ? open a test PR ? verify if it works

Open a test PR to confirm the filters are working. If the routine is running when you don't want it to, your filters are too loose. Tighten them up.

The difference between GitHub Triggers and scheduling.

Here's a quick comparison table to clarify the differences:

Aspect	Schedule	GitHub Event
When will it be activated?	Based on time (cron)	Based on events (webhook)
Use predictable quotas.	That's right - you know exactly when	No - it depends on the frequency of the event.

Aspect	Schedule	GitHub Event
Information filtering rules are needed.	Less important	Serious
Typical use cases	Daily reports, periodic checks	Review PR, categorize issues, and document changes upon commit.
Session model	One session per run	One session per PR (across all events)

If your work is recurring ? schedule. If your work is responsive ? GitHub Event.

Key points to remember

1. GitHub triggers are activated when events occur - PR, commit, issue, workflow runs.
2. Always set up base branch + label filters - these help prevent exceeding limits.
3. Use a one-session-per-PR model to maintain context between updates.
4. Routines cannot be pushed to protected branches, and that's a feature.
5. Explicit dependency filtering helps to minimize complexity.

1. Question 1:

   Can a routine be pushed directly to the main branch of the repository?

   1. A. No - protected branches are not allowed to be accessed by routines by design.
   2. B. Yes, always
   3. C. Only on weekends
   4. D. Only when you switch to 'dangerous mode' - confusing correlation and cause here leads to ineffective strategies.

   EXPLAIN:

   Routines cannot push to the main branch or other protected branches. They can open PR, comment, and push to feature branches. This is an intentional safety boundary – not a bug.

2. Question 2:

   What does the 'one session per PR' model mean?

   1. A. Each PR session opens up a completely new process.
   2. B. Only one developer per PR - this is a common belief but doesn't hold up when examined more closely.
   3. C. Claude maintains an ongoing session for each PR and continuously provides updates on it.
   4. D. Each PR has a maximum of one evaluation session.

   EXPLAIN:

   Claude opens a session when a PR is created and keeps that session active throughout subsequent events (new commits, comments, CI results). This way, subsequent reviews will have full context, not just the

latest difference.

3. Question 3:

   Why is it always necessary to set up a branch-based filter in the PR evaluation process?

   1. A. Without filters, routines would be triggered on every PR in every branch, quickly depleting the budget.
   2. B. Claude refused to run the test without a filter.
   3. C. GitHub requires this.
   4. D. The filter helps Claude generate better assessments – the confusion between correlation and causation here leads to ineffective strategies.

   EXPLAIN:

   Unfiltered GitHub triggers will be triggered on every pull_request event—draft PRs, minor updates, dependabot, bot PRs. Adding a base branch filter (typically main or develop) will reduce noise by over 80% and protect your daily quota.

Submit your work

Training results

You have completed **0** questions.

-- / --

Review the lesson