# RFC 227: Code Monitoring

Editor: Quinn Keast
Status: Approved
Requested reviewers: Pooja, Christina, Rob, Alicja, Keegan, Stephen, Chayim, Farhan, Dan
Approvals: Chayim, Pooja, Farhan, Joe

## Background

As part of PD 12: Saved Searches / Search Notifications, we're untangling and tackling two separate product features: "saved searches" as a way of saving useful searches for future reuse and sharing, and "code monitoring and notifications/actions" as a way to use search queries to trigger notifications or external actions for a variety of use cases.

This RFC is intended to explore a high-level conceptual approach to **code monitoring and actions/notifications**, which will then be used as the basis for design exploration.

The other half of this problem is explored in RFC 226: Improve Saved Searches and Uncouple from Notifications.

## Problem

PD 12 puts forward quite a few challenges with the current state of our saved searches/notifications feature, which I've summarized for a quick overall understanding.

As it relates specifically to "saved search notifications," or rather "code monitoring and actions/notifications,", we have these challenges:

**Problem statements**

- We've observed that the "saved search" notification emails Sourcegraph sends today create an extra alerting system for users to consume that doesn't provide the benefits already present in their existing alerting systems, which creates more "noise" without contributing to existing UI metrics or creating positive social pressures. If we expand the types of notification channels available to users to align with the channels they're already using, it will make "saved search" notifications more valuable and usable for our users.

- We've observed that it's difficult for users to control who is notified in response to "saved search" notifications, which creates more work for team members who have to share notifications and information with their team manually. If we improve this, we will make "code monitoring and notifications" a natural and invaluable part of the broader development and compliance workflow.

- We've observed that the information provided in "search notification" emails today is not intuitive or helpful for users without their already knowing the context of the notification, which makes notifications harder to consume and act upon. If we improve this, we'll make notifications more immediately actionable and valuable for our users.

- We've observed that the functionality, purpose, and value of "saved searches" in relation to notifications isn't clear or intuitive in the product, which makes it less likely our users will explore and adopt the feature. If we improve this, we'll make it easier for users to understand the purpose and benefits of code monitoring and actions/notifications.

**Top problems**

- The way "saved searches" are implemented today is not intuitive and blurs the purpose of the feature by combining the separate concepts of "saved search for easy re-use" and "code monitoring and notifications," which makes it less obviously useful for users.
- The current "saved search" page is hard to use as a place to check on things, which creates friction.
- The data provided in the current email notifications is not helpful without existing knowledge and context, which makes notifications hard to share and consume.
- With email as the only notification channel, the potential value for users with existing alerting systems is lost.
- Notifications can't be consumed on Sourcegraph itself, which makes notifications hard to monitor and rely upon.
- The search types available for "saved search" queries that can result in notifications are limited to diffs and commits, but this isn't clearly communicated, which makes the workflow for defining "saved search notifications" confusing.
- Notifications are sent too frequently without control over recipients or timing, which creates "noise."
- Setting up email in Sourcegraph is a painful experience for users and admins, which adds a lot of friction up front.

# Proposal

## Establish code monitoring and actions/notifications as a distinct feature in the UI

I strongly believe we have to uncouple the concepts of "saved searches for easy re-use" and "code monitoring and notifications." In a separate RFC, we're exploring how we might make "saved searches for easy re-use" more intuitive and useful.

The first step is to expose code monitoring and notifications/actions as a distinct feature, by adding a new section for `Code monitoring` in the user and organization profile.

> **Sidebar: "Search monitoring" vs "Code monitoring"**
> "Search monitoring" has double meaning. The implicit first meaning is "monitoring searches carried out by users," rather than "monitoring when the result of a search changes." It's more accurate to describe this as "Code monitoring" – it's changes to the code, some of which we *detect* through search queries, that actually trigger actions and notifications. As well, "search monitoring" conceptually limits what can trigger actions or notifications.

Then, we should add an entity called a "code monitor."

A monitor has:

- Name
- Description
- Ownership
    - Personal
    - Organization
- Enabled/Disabled
- A trigger/event [1]
    - New search result
    - (Future search results)
- One or more actions [2]
    - An email notification
    - A webhook
    - A Zapier event
    - Etc.

A monitor can be created for the user's personal use, or shared within an organization.

The trick to this that would make it sustainable for pretty much everything in the future is the combination of triggers/events and actions.

## [1] Triggers/events

Today, we have "saved search notifications." A search query is used to target search results. When email notifications are enabled, "Sourcegraph [will] automatically run your saved searches and notify you when new results are available via email."

There's three problems with this: first, it combines the concept of code monitoring and notifications with the concept of saved searches for easy re-use. Second, it's not clear exactly what is being monitored and what triggers exist. Finally, it limits what can be a trigger to the results of a search query, which limits future product opportunity.

With GitHub actions and webhooks, there are clearly defined events like `push`, `pull request`, or `issues`. Similarly, GitHub notifications have specific triggers like being `@mentioned`, comments being posted, or for Dependabot, when a new vulnerability is found.

To make code monitoring and notifications intuitive, we need to expose the concept of a trigger or event.

In the short term, our first trigger will be something like **`new search result`**, which is targeted using a search query. This just reframes the current "query" into a trigger event.

In the future, we may introduce other trigger events that makes the feature even more useful for our users.

## [2] Actions

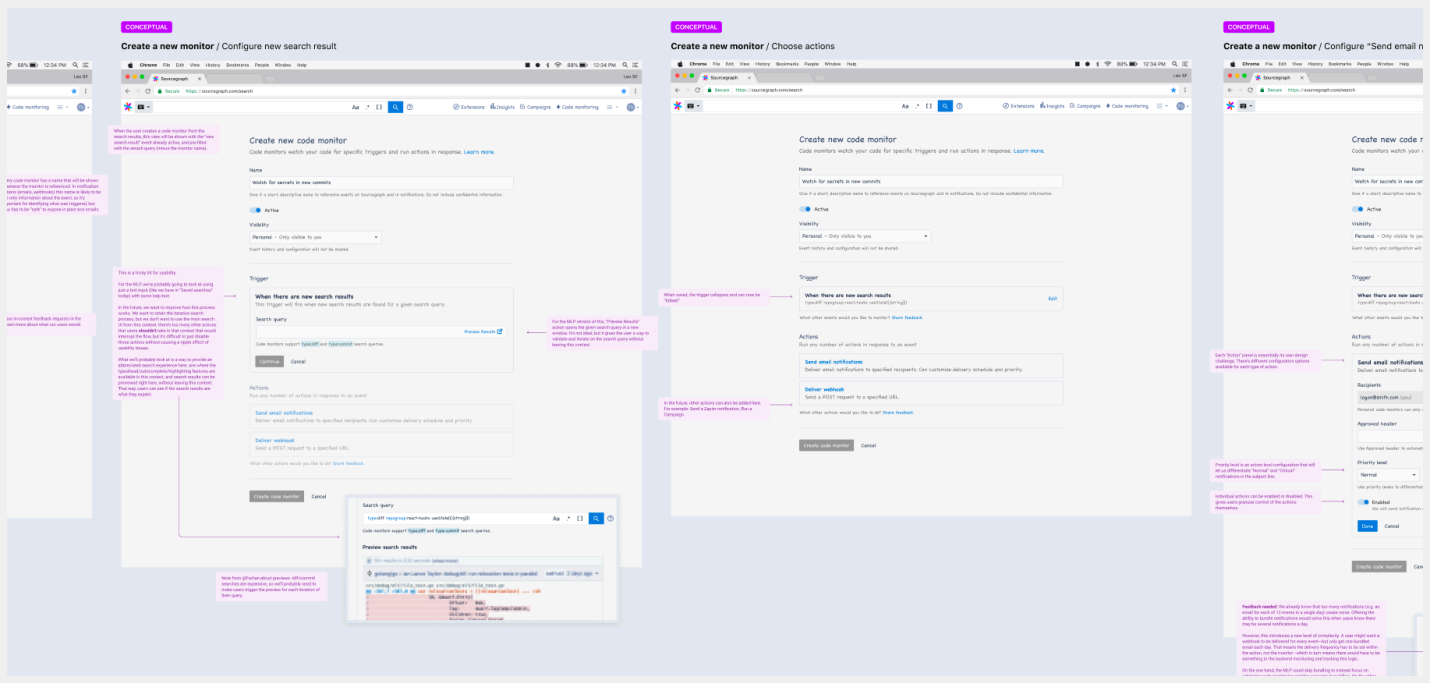
What users can do in response to a trigger is a huge product opportunity area. Today, we only offer email notifications, but there's an appetite to offer other channels like webhooks. If we design for the future, we'll make it easy to expand the value of the feature in the future—without investing too much effort up front.

Since each channel has a slightly different purpose and setup, we would first let the user choose which actions they want to happen in response to the trigger, and then let them configure each action independently. Users can choose to have more than one action run in response to a trigger.

An action has independently configurable preferences. For example, an email notification as an action has timing or bundling options. A webhook has SSL verification.

## Initial design exploration

Design and product have done an initial exploration into how the code monitoring concept could be supported. The Figma link provided here is for an annotated wireframe exploration, and contains all additional context needed to understand the implementation.




**Please share your thoughts, questions, or concerns as Figma comments!**

- View on Figma

## How code monitoring fits in with campaigns

The campaigns team identified that there are potential overlaps between code monitoring as proposed in this RFC and campaigns.

- Campaigns do not run automatically in the background today, but they will be.
- A campaign spec could be extended to allow for things like sending email or slack notifications.
- A campaign could run automatically in response to new search results.

I propose that campaign and code monitoring solve different baseline use cases, and rather than overlapping in their solutions, complement each other.

Campaigns are for *making changes across a codebase*, while code monitoring is for *monitoring for events and running actions in response*.

This aligns with what we've heard about some use cases: users sometimes want to take actions that do not involve immediately starting a campaign. Instead, they want to be notified of important code changes first, and then decide which follow up action(s) they need to take. Some examples:

- Customers want to notify the PR author of the code change and give them X days to revert their change
- They want to forward these notifications to their internal security team for review
- They would like to integrate these notifications into their existing channels (e.g. pager duty) where they review which code monitor results are severe enough to take action on.

While notifications and webhooks could be defined in a campaign spec, it moves a lot of responsibility and configuration into the YAML, which creates a barrier to adoption in complexity and also implies that notifications are only in response to a campaign's execution.

But, a campaign could run in response to an event (or even trigger an event). We already plan to explore other ways of triggering a campaign, such as when a changeset (pull request) is merged, or when a CI check has finished running, etc). Running a campaign as an action in code monitoring would just be another way to take advantage of campaigns, while code monitoring itself would provide immediate value outside of the context of campaigns.

Take, for example, the case of monitoring and responding when secrets like AWS secret keys are committed to a codebase.

- A campaign can be pre-defined by a security team to automatically propose a PR to remove the secrets from the code.
- A code monitor can be created to monitor for secrets committed to a codebase, and run a series of relevant actions in response: `Run a campaign`, `Send email notifications`, and `Deliver webhook`.

**Moving forward in the short term**

For the MLP for code monitoring, we want to validate the underlying concepts and use cases. Along the way, we're thinking about how code monitoring and campaigns can support each other without introducing blurry edges around the purpose of each feature.

# Project process

1. Get approvals on this RFC to use as a starting point.
2. Explore this proposal in design.
3. Create prototype and test internally; iterate.
4. Hold sessions with customers that express interest in monitoring and notifications to evaluate.
    a. Will be reaching out to customers to evaluate concept.
5. Create MLP implementation with limited feature set (1 trigger, 1-2 actions).
6. Validate and explore other useful actions that customers may use.

## Definition of success

We'll know this project is working if we see:

- Increased adoption of code monitoring
    - Total # of users who have a "code monitor"
    - Total # of users who interact with a "notification"
- Increased engagement with code monitoring and notifications/actions
    - Average # of monitors per user
    - Average # of "notification" interactions per user
- Increased engagement with Sourcegraph due to code monitoring notifications/actions
    - Total # of interaction events with notifications/actions
    - Ratio of # of notifications interacted with vs. # of notifications sent
- Happiness with code monitoring and notifications/actions feature
    - Average happiness score in relation to feature *(this doesn't exist today, we'd have to create it)*

## Initial engineering insights

Technical constraints, assumptions, and remarks from PD 12: Saved Searches / Search Notifications

- Support for `type:diff` and `type:commit`
    - Assumption is `type:diff` and `type:commit` search is required and sufficient for the primary use case of code monitors which is to notify users when new code is introduced to the code base.
    - Note that `after:` and `before:` used for aggregating search results does not work in search modes other than `type:diff` or `type:commit`: Issue
- Saved searches are not subject to any 50 repository limit. This does not scale when search triggers return sets of code results that are too large.
    - From Stephen: "If the before/after time range we query at once is too large and the repository is too large it means the search may not complete, would timeout, and the user and site admin would not be notified there is a problem at all. One such example of this is that this query returns a result but if you change "after:"1d ago" to "10d ago" you get no results: https://sourcegraph.com/search?q=repo:%5Egithub%5C.com/torvalds/linux%24+type:diff+after:%2210d+ago%22+xio&patternType=literal"
    - Performance work needed for scheduling notifications: Issue.
        - Stephen's suggestion on frequency of scheduling: "changing the run time to every 24h is easy to do and would be a good first step I believe."
- Security concern for content shown in monitor/notification
    - From Stephen "We will need to be extremely careful with providing *any* content other than a link in the actual notification itself. We don't know that the person receiving the notification actually has permission to view the content": Issue
    - Additional info from Chayim: "In general, there's a security goal to not expose information. Information exposure is a big problem, especially combined with the fact that email (i.e the notification side) is plain text."
- Investigation on supporting notification channels from Chayim
    - Mail servers

        - SMTP fixes,  AWS Simple Email Service (Issue, Issue, AWS docs): AWS SES can be used as an SMTP server, so there's no need to call it out unless we want to support direct integration with Amazon credentials.
    - Webhooks
        - A nice generic way of transmitting data via HTTPS. Many services (including Slack) consume them: Issue
    - Notification ingestion from customers
        - Important for security vertical, required for legacy security SIEMs that rely heavily on collecting syslog, solution is outputting to syslog. SIEM integration means supporting two export formats on our side:
            - syslog - SIEMs generally import unstructured log events if they're in the syslog format . We would have to publish a format document accordingly so that customers could configure their SIEMs.
            - CEF - Many SIEMs also support the Common Event Format. By support CEF output we make direct SIEM integration a better experience for customers - but unfortunately only in the case of using a SIEM (Splunk, Arcsite, etc) that supports CEF.

## Constraints, remarks, and insights

- “Will we introduce an in-platform notification system?”
    - Concerns around scope and notification fatigue.
    - Would likely support notifications from more than just code monitoring.
    - For the MLP, no. Will evaluate in the future.
- “Do we have any data around how many companies use our organizations feature?”
    - From Dan: “I've never once heard a user/customer/prospect mention using orgs, and I've never recommended it to anyone outside of the saved search context. The value you get from orgs (shared settings, shared saved searches) is pretty minimal, and requires knowledge of pretty advanced features, as well as manual management (versus pulling in groups from SSO or whatever). Even without having specific ping data behind this, I feel extremely confident that they're very, very, very rarely used.”
- “This might be out of scope for this RFC, but how would you define "new search result?"If the search query yields 5 matches in 5 files in 5 repositories — will the action be triggered when anything in that result set changes (less results, more results, *exact* matches change)?”
    - To be addressed in design.
- “Potential triggers in the future?”
    - From Pooja: A few potential “extension-based” search trigger use cases have come up recently. We can start capturing these in a research section as we learn of more of these.
        - Customer: https://app.hubspot.com/contacts/2762526/company/554275594/
          When test code coverage (available through the Codecov extension) drops below 60%, they would like to receive email notifications.
        - Customer: https://app.hubspot.com/contacts/2762526/company/557692623/
          Use case: “JFrog Xray extension pipes in results that ultimately trigger Sourcegraph campaigns to resolve security issues related to licenses and vulnerabilities”
          The extension returns results that are security issues it scans for. The results can be potential search triggers that are used to start a campaign.