Sourcegraph

Cody

Security and legal whitepaper

This document was last updated on August 18th, 2023. Cody remains a Beta product and we will continue to update this document as we ship product updates.

Sourcegraph Cody ("Cody") is a state-of-the-art AI coding assistant designed to help software developers by answering code questions and generating code. Cody uses Large Language Models (LLMs), Sourcegraph search, and Sourcegraph code intelligence to provide accurate and context-specific results to developers, streamlining their workflow, reducing the time spent on repetitive tasks, and improving the quality and consistency of their work.

Cody can be enabled on demand on an organization's Sourcegraph instance. Once Cody is turned on in an Sourcegraph instance, any user can configure and use the Cody VS Code extension - this does not require admin privilege. Cody is designated an experimental feature because we expect the product capabilities and user experience to change rapidly.

This whitepaper covers the critical aspects of Cody's integration into enterprise environments, including security and legal considerations. Each section covers the current state, future state, and key discussion topics.

For information related to Cody features, usage, and administration, see the Cody product page and Cody documentation. For general information about Sourcegraph security, including our SOC 2 report, visit the Sourcegraph security portal.

Architecture and data flow

Developers use Cody in the editor and in their organization's Sourcegraph instance's web application. When developers use Cody, Cody communicates with the organization's Sourcegraph instance and with a Large Language Model (LLM) API to return information to the user.

Components

Cody currently consists of 4 components:

- Cody client
- Sourcegraph instance
- Large Language Model (LLM) API
- Embeddings API

Cody client

The Cody client is the application used by the user to interact with Cody, such as the Cody VS Code extension or certain parts of the Sourcegraph web application.

Sourcegraph instance

The Sourcegraph instance is an organization's Sourcegraph Enterprise server, either deployed self-hosted within the organization's network or in a dedicated single-tenant Sourcegraph Cloud environment, or Cody App for individuals using Cody outside the enterprise.

Large Language Model (LLM) API

The Large Language Model (LLM) is the model that accepts prompts and returns completions, consisting of an answer or generated code. Cody currently uses Anthropic Claude as the preferred LLM.

Embeddings API

Embeddings enhances response quality in some cases. This feature must be explicitly enabled by an administrator. It currently relies on OpenAI's embeddings API. We plan to implement an Sourcegraph Managed embeddings generator to remove this dependency on an external provider. Until then, OpenAI is the available solution for customers.

Data flow

There are 2 types of data flows:

- Interactive usage
- Embeddings generation

Interactive usage

Interactive usage is when a user types a message to Cody or otherwise performs an explicit action in their editor (such as "Ask Cody > …") that types a message to Cody on their behalf.

Components involved

- Cody client
- Sourcegraph instance (code context and completions API endpoints)
- LLM API (completions API endpoint)

Data involved

- User query
- Code context (code and text file contents and other metadata)
- Request header data (authentication and user identity)

Data retained

- Audit logs of user events are retained on the Sourcegraph instance and any external logging destinations, according to the Sourcegraph site configuration.
- No data (neither LLM inputs nor outputs) is retained by the LLM API or Embeddings Service.

Process

1. To respond to a user request, the Cody client communicates with the Sourcegraph instance's code context API to fetch code context that is relevant to the query.
    a. For example, if the user asks Cody "What is the `foo` API?", the code context would include files that the Sourcegraph instance determines are relevant to `foo`.
    b. Permissions are enforced by Sourcegraph, so that users may only query in repositories or receive file contents that their user account is permitted to view on the Sourcegraph instance.
2. The Cody client concatenates the user request and the code context into the prompt.
    a. For example, if the user asks Cody "What is the `foo` API?" and a single file `foo.java` is deemed relevant to the query, the prompt would resemble the following simplified example: "<contents of foo.java>\n\nWhat is the `foo` API?".
3. The Cody client sends the prompt to the Sourcegraph instance's completions API.
4. The Sourcegraph instance's completions API forwards the prompt to the LLM's completions API.
5. The LLM's completions API returns completions to the Sourcegraph instance, which passes it back to the Cody client.

Embeddings generation

Embeddings must be explicitly enabled by an administrator.

Components involved

- Sourcegraph instance (embeddings queue and blob store)
- Embeddings API (generation API endpoint)

Data involved

- Repository contents (with configurable exclusion patterns via site config)
- Generated embeddings data

Data retained

- Audit logs of the enqueue request and of the job status are retained on the Sourcegraph instance and any external logging destinations, according to the Sourcegraph site configuration.
- The embeddings data structure is stored on the blob store (such as a local file system or Amazon S3) configured in the Sourcegraph site configuration.

Process

1. When a Sourcegraph site admin enqueues a repository for embeddings generation, Sourcegraph iterates through each file in the repository (in batches) and sends their contents to the external embeddings API's generation endpoint.
    a. Site admins may configure a set of paths and path patterns to exclude from being sent to the embeddings API.
2. The embeddings API returns a data structure used to perform similarity search on the repository's contents.
3. The data structure is serialized to an object that is stored in the blob store (such as a local file system or Amazon S3).

Data security, retention, and compliance

Cody relies on Sourcegraph, which is SOC 2 compliant. When Cody is enabled and used, there are 3 components that interact with sensitive data:

- Cody client
- Sourcegraph instance
- Large Language Model (LLM) API

Data security, retention, and compliance is discussed in the context of each component in the following sections.

Cody client

The Cody client is the application used by the user to interact with Cody, such as the Cody VS Code extension or certain parts of the Sourcegraph web application. This section only refers to official Sourcegraph Cody clients, not any current or future unofficial clients.

All Cody clients respect the repository permissions configured on the Sourcegraph instance. Cody never uses any information from unauthorized repositories to respond to a user. In some cases, due to fundamental limitations of LLMs, Cody may falsely claim to have knowledge of repositories that the user is not permitted to view. For example, if the user asks Cody "What does <secret repository> do?" (where "<secret repository>" is a repository that the user is not permitted to view), Cody may respond with a false but confident-sounding response ("hallucination") that is not based on any actual knowledge of the unauthorized information and thus does not present a security risk. Users can confirm that Cody did not, in fact, access the claimed unauthorized repositories by viewing the list of code context (if any) that Cody actually consulted in the Cody UI.

When using Cody via the VS Code extension, Jetbrains extension and the Sourcegraph web application:

- The transcript of user-Cody interactions is stored in memory only and is not persisted.
- The VS Code extension requires the user to specify the following in VS Code user or workspace settings: the Sourcegraph URL, the name of the repositories to consult, and other non-sensitive user preferences.
- The VS Code extension stores the user's Sourcegraph access token in the VS Code secrets store, which uses the operating system's standard secure keychain for secret storage.
- No other data is stored or retained on the client.

Sourcegraph instance

The Sourcegraph instance is an organization's Sourcegraph Enterprise server, either deployed self-hosted within the organization's network or in a dedicated single-tenant Sourcegraph Cloud environment. If you are using Sourcegraph as an individual, this will be via Cody App running locally.

A user's Cody client accesses the Sourcegraph instance using an access token, which the user generates on the Sourcegraph web application in their user settings area. Administrators may view, audit, and revoke user access tokens at any time.

The Sourcegraph instance records audit logs of user interactions with Cody, if the administrator has enabled audit logs in the site configuration. These audit logs may reside on the Sourcegraph instance and/or may be sent to a custom log destination if you integrate your cluster with external log forwarders.

No other data is stored or retained on the Sourcegraph instance as a result of Cody usage.

(As noted above, this section assumes that embeddings has not been explicitly enabled by the administrator.If you have discussed using that feature with the Sourcegraph team and intend to explicitly enable it, you can find the embeddings documentation here.)

Large Language Model (LLM) API

The Large Language Model (LLM) API accepts prompts and returns completions, consisting of an answer or generated code.

Cody currently uses Anthropic's Claude as the preferred LLM. Cody sends requests from the Sourcegraph instance to Anthropic's API, which resides at https://api.anthropic.com. To reduce the risk of data leakage and compliance concerns, Cody Enterprise offers :

- Zero data retention: Prompts (inputs) and results (outputs) are not retained or stored by Anthropic. Prompts are only processed for the duration needed to return the results and are permanently removed afterwards.
- No other data sharing: Prompts and results are not shared with any other third parties for any purpose at any time.
- No training based on your data: Prompts and results are not used to customize or train Anthropic's models.
- No PII: Anthropic does not receive any personally identifying information in the request headers or metadata about the end user.

These guarantees are currently stronger than those from any other vendor's competitive offering.

If an administrator explicitly configures the Sourcegraph instance to use a different LLM API (such as OpenAI or a self-hosted API-compatible LLM), we are currently unable to offer any data security guarantees.

Model provenance

Today's leading Large Language Models (LLMs), such as Anthropic Claude (the default LLM for Cody) and OpenAI GPT-4, are trained using a diverse range of publicly available data. This is an industry-standard approach to training AI models, which ensures that the model is well-equipped to handle a wide variety of tasks and domains. However, there may be cases where the LLM inadvertently generates code that is similar to code found in its training data.

To mitigate this risk, Sourcegraph intends to offer indemnification to enterprise customers for intellectual property infringement caused by Cody. We are also working on developing tooling for identifying potentially copied code. This will provide customers with the confidence that the generated code complies with legal requirements and reduces the likelihood of intellectual property infringement.

By continuously improving and refining our models and tools, we aim to provide a safe and reliable coding assistant to software developers, while also respecting the intellectual property rights of others in the industry.

Discussion topics

We are evaluating the following future improvements to Cody related to security and legal requirements. Please discuss your organization's specific needs with us to get more information about these possible future improvements.

- Better support for self-hosted LLMs
- Support for more LLM APIs
- Ability to train and fine-tune models on custom datasets of code and documentation
- Support for embeddings generation that does not depend on an external provider
- Model provenance
- Enforcement of coding standards and criteria in generated code
- Insecure code detection (in generated code and existing code)
- Tracking when developers insert Cody suggestions into the codebase (with transcripts, history, etc.)
- Fine-grained restrictions on Cody usage on certain repositories, paths, file categories, users, etc.
- Filtering system for suggestions to omit insecure code, and fake-but-realistic-looking PII