Block suggestions matching public code

To block suggestions from GitHub Copilot that match public code, you can configure specific settings either as an individual user or at the organization level if you have the necessary administrative rights.

Individual User Settings

1. **Access Your GitHub Settings**:
   - Click on your profile photo in the upper-right corner of any GitHub page and select "Settings".
   - In the left sidebar, click "Copilot".
2. **Configure Suggestions Matching Public Code**:
   - Under "Suggestions matching public code", select the dropdown menu and choose "Block" to prevent Copilot from suggesting code that matches publicly available code.
   - Click "Save" to confirm your new settings.

When this setting is enabled, GitHub Copilot will check code completion suggestions and their surrounding code (about 150 characters) against public code on GitHub. If a match or near match is found, the suggestion will not be shown to you.

Organization-Level Settings

1. **Access Organization Settings**:
   - Go to your profile photo, select "Your organizations", and choose the relevant organization.
   - Click on "Settings" and navigate to "Copilot" under the "Code, planning, and automation" section.
2. **Set Policies**:
   - Under "Policies", find the "Suggestions matching public code" dropdown menu and select "Blocked".
   - This will apply the setting to all organization members who use Copilot.

When enabled, this setting helps prevent potential copyright issues by ensuring that Copilot does not suggest code that is identical or similar to publicly available code on GitHub. This measure is particularly useful for maintaining code originality and complying with licensing requirements

Additional Information

GitHub Copilot includes a code referencing feature that can detect and suppress suggestions that contain code segments matching public code on GitHub. This filter can be enabled by enterprise administrators and applied across all organizations within an enterprise. The filter checks for matches or near-matches against public code, and if a match is found, the suggestion is blocked. This helps mitigate the risk of using code that might have legal or licensing implications

Exclude specified files from Copilot

To exclude specific files from being used by GitHub Copilot, you can configure content exclusions at both the repository and organization levels. Here’s how you can do it:

Repository-Level Exclusions

1. **Access Repository Settings**:
   - Go to the settings of your repository.
   - Navigate to the "Code & automation" section and select "Copilot".
2. **Specify Paths to Exclude**:
   - In the "Paths to exclude in this repository" text box, enter the file paths you want Copilot to ignore.
   - Use the format: `- "/PATH/TO/DIRECTORY/OR/FILE"`.
   - You can use fnmatch pattern matching notation, which is case insensitive.

Example (YAML):

```yaml
- "/src/some-dir/kernel.rs"
- "secrets.json"
- "secret*"
- "*.cfg"
- "/scripts/**"
```

Organization-Level Exclusions

1. **Access Organization Settings**:
   - In GitHub, select your profile photo, then click "Your organizations".
   - Choose the organization, go to Settings, and then select "Copilot".
2. **Enter Paths for Exclusion**:
   - In the "Repositories and paths to exclude" box, specify the repositories and the paths within those repositories that should be excluded.

- Use the format (YAML):

  ```
  REPOSITORY-REFERENCE: - "/PATH/TO/DIRECTORY/OR/FILE"
  ```

  REPOSITORY-REFERENCE could be the reponame within the orgranization or the https link

Example (YAML):

```
octo-repo:
        - "/src/some-dir/kernel.rs"
        - "secrets.json"
https://github.com/primer/react.git:
        - "/src/**/temp.rb"
```

Using .copilotignore File

For VSCode, you can use the `.copilotignore` file to specify patterns of files and directories to exclude. This is similar to `.gitignore` but specifically for Copilot.

Example:

```
# Ignore all environment files
**/.env

# Ignore all files in the scripts directory
/scripts/**
```

By setting these exclusions, you ensure that GitHub Copilot does not use the specified files to provide code completion suggestions, which can be particularly useful for sensitive or irrelevant files

Organization-wide policy management

As an organization owner on GitHub, you can manage various aspects of GitHub Copilot to tailor its use to your organization's needs. Here are key features and steps to effectively manage these policies:

Setting Policies

1. **Access Organization Settings**:

    - In the upper-right corner of GitHub, click your profile photo and select "Your organizations".
    - Choose the relevant organization and click "Settings".
2. **Enable or Disable Copilot Features**:

    - Navigate to "Code, planning, and automation" and select "Copilot", then "Policies".
    - Here, you can enable or disable features like:
        - Copilot Chat in IDE
        - Copilot Chat in GitHub Mobile
        - Copilot in the CLI
        - Suggestions matching public code

    To manage public code suggestions, select the "Suggestions matching public code" dropdown and choose "Allowed" or "Blocked". Blocking suggestions ensures that any code completions resembling public code on GitHub won't be shown

Granting and Managing Access

1. **Grant Access to Copilot**:
    - In the organization settings, go to "Copilot" and then "Access".
    - To enable Copilot for all members, select "Enabled For: All members of the organization" and confirm the seat purchase.
    - To enable Copilot for specific users, select "Enabled For: Selected members" and assign seats individually or in bulk via CSV upload
2. **Manage Requests**:

- Approve or deny access requests in the "Requests from members" section under the "Access" settings. This helps streamline who can use Copilot within your organization

Additional Controls

- **Content Exclusion**:
    - Exclude certain files or directories from Copilot suggestions by configuring paths in the "Content exclusion" section. This is useful for keeping sensitive or irrelevant files out of Copilot's suggestion pool.
- **Audit and Monitoring**:
    - Regularly review audit logs and user activity data to monitor how Copilot is being used within your organization. This helps in making informed decisions about seat assignments and ensuring compliance with organizational policies

Reviewing Audit Logs for GitHub Copilot

As an organization owner, you can utilize audit logs to track and review actions related to your GitHub Copilot Business subscription. Here’s how you can manage and access these logs:

Accessing Audit Logs

1. **Navigate to Settings**:
    - In the upper-right corner of GitHub, select your profile photo and click "Your organizations".
    - Choose the relevant organization and click "Settings".
2. **Viewing Logs**:
    - In the sidebar, under "Archives", click "Logs" and then "Audit log".

Searching Audit Log Events

You can search for specific events within the audit logs using various qualifiers:

- **By Action**: `action:copilot` returns all GitHub Copilot audit log events.
- **By Operation Type**: Use qualifiers like `operation:access` to find access-related events.
- **By Repository**: `repo:my-org/our-repo` limits results to a specific repository.
- **By User**: `actor:username` filters events performed by a specific user.

Copilot-Specific Events

Audit logs can track various Copilot-related actions, such as:

- Changes to Copilot settings and policies.
- Addition or removal of seats from your subscription.
- Updates to the duplicate detection policy.
- Seat assignment changes.
- Content exclusions updates.

These logs provide detailed information including the action performed, the user who performed it, the affected repository, and the date and time of the event.

Using Audit Logs for Monitoring

The audit logs cover events for the last 180 days and include detailed entries about:

- **Seat assignments**: Tracking when seats are assigned or revoked.
- **Policy updates**: Monitoring changes to policies at the organizational level.
- **Content exclusions**: Viewing updates to paths excluded from Copilot suggestions.

For a full list of possible events and detailed instructions on using audit logs, you can refer to GitHub's documentation on reviewing audit logs for Copilot Business and audit log events.

By leveraging these audit logs, you can maintain transparency and control over the use of GitHub Copilot within your organization, ensuring compliance with your internal policies and standards.

Copilot Chat skills

GitHub Copilot Chat offers a range of skills to assist developers in their coding tasks through an interactive chat interface available on GitHub.com, supported IDEs, and GitHub Mobile. Here's an overview of the skills and functionalities provided by Copilot Chat:

Core Skills

1. **Answering Coding Questions**:
   - You can ask Copilot Chat for explanations on specific coding problems, general software questions, or details about your project. For instance, you can inquire about different frameworks, how to implement specific features, or ask for explanations of your existing code.
2. **Writing Code**:
   - Copilot Chat can generate code snippets based on your prompts. You can request it to write new functions, add error handling, or even create entire modules. For example, you might ask it to write a function that sums all numbers in a list or to set up a new project with specific frameworks.
3. **Fixing and Improving Code**:
   - It can suggest fixes for bugs, optimize code performance, or refactor code to improve readability and maintainability. You can use specific commands like `/fix` to propose solutions for errors in the selected code.
4. **Explaining Code**:
   - By selecting a block of code and asking Copilot Chat, you can get natural language descriptions of what the code does, its input/output parameters, and its role in the larger application. This feature is useful for understanding complex code sections or explaining them to non-technical stakeholders.
5. **Generating Tests**:
   - Copilot Chat can generate unit tests for your code. Using the `/tests` command, you can ask it to create tests using frameworks like Jest, ensuring your code

functions as expected and catching potential issues early.

6. **Setting Up Projects**:
    - It can help set up new projects by suggesting directory structures and creating the necessary files. For example, you can ask it to set up a new React app with TypeScript or a Node.js Express server.

Copilot pull request summaries

GitHub Copilot Pull Request Summaries is an AI-powered feature designed to streamline the code review process by automatically generating summaries for pull requests. This feature is currently available exclusively to GitHub Copilot Enterprise users.

Key Features of Copilot Pull Request Summaries

1. **Automatic Summarization**:
   - When a pull request is created or updated, Copilot can generate a summary that includes a high-level overview of the changes and a detailed outline of the modifications. This helps reviewers quickly understand the scope and impact of the changes without manually sifting through each file and line of code

2. **Summary Components**:

- The summary typically consists of two parts:
    - **Prose Overview**: A paragraph summarizing the key changes in the pull request.
    - **Bulleted List**: An itemized list of changes linked to the specific lines of code affected, providing precise context for each modification

3. **Enhanced Review Efficiency**:

- By providing structured summaries, Copilot helps reviewers focus on critical parts of the code, potentially increasing the efficiency and effectiveness of the review process. This can be particularly useful for large pull requests or when dealing with complex codebases

How It Works

- **Pipeline Process**:
    - When a summary is requested, Copilot initiates a workflow that analyzes the code diffs and generates the summary using a large language model. This process includes parsing the raw diffs and creating a prompt that the model uses to produce the summary

- **Usage**:
    - You can initiate a summary when creating a pull request, by editing the pull request description, or by adding a comment in the pull request thread. Feedback mechanisms allow users to rate the quality of the summaries, which helps improve the feature over time
    -

Limitations

- **Scope and Accuracy**:
    - The feature currently supports English and may have limitations in handling very large pull requests or accurately capturing every nuance in complex changes. Users are advised to review the summaries for accuracy and completeness

Best Practices

- **Supplemental Tool**:
    - While Copilot Pull Request Summaries can save time, it should be used as a supplement rather than a replacement for detailed human reviews. Reviewers should add additional context and validate the AI-generated summaries to ensure a comprehensive understanding of the changes

Copilot knowledge bases

GitHub Copilot Knowledge Bases is a feature available for GitHub Copilot Enterprise users that enhances the AI's ability to provide contextualized assistance by leveraging your organization's internal documentation. Here’s an overview of how to use and manage this feature:

Creating and Managing Knowledge Bases

1. **Permissions**:
    - To create and manage knowledge bases, you need to have administrative rights within your GitHub organization.
2. **Creating a Knowledge Base**:
    - Navigate to your organization's settings on GitHub.
    - Go to the Copilot settings section and select the option to configure knowledge bases.
    - You can create a knowledge base from one or multiple repositories. This can include public, private, and internal repositories. Ensure the repositories contain

relevant Markdown documentation.

3. **Configuring Knowledge Bases**:
    - Name your knowledge base and optionally provide a description.
    - Select the repositories you want to include by checking the appropriate boxes and then apply your selections.
4. **Assigning Knowledge Bases to Users**:
    - Organization members can use these knowledge bases as context for Copilot Chat in GitHub.com and supported IDEs like VS Code. When asking a question, Copilot will search the specified knowledge base for relevant information to generate a response .

Usage and Benefits

- **Contextual Assistance**:
    - When you ask a question in Copilot Chat, specifying a knowledge base ensures the AI uses your internal documentation to provide more accurate and relevant answers. This is particularly useful for complex projects where internal best practices and specific coding standards are documented.
- **Increased Productivity**:
    - By leveraging knowledge bases, developers can get quick, context-aware responses to their queries, reducing the time spent searching for information and increasing overall productivity.
- **Enhanced Collaboration**:
    - Knowledge bases help ensure all team members have access to the same information, promoting consistency and better collaboration across projects.

Best Practices

- **Regular Updates**:
    - Keep your knowledge bases updated with the latest documentation to ensure the information Copilot provides is current and accurate.
- **Feedback Loop**:
    - Use the feedback options available in Copilot to continually improve the quality of responses generated from knowledge bases. This can help refine the AI's understanding and utility over time.

For more detailed instructions and best practices, you can refer to GitHub’s documentation on managing Copilot knowledge bases here

Zero data retention for code snippets and usage telemetry

GitHub Copilot has implemented a zero data retention policy for code snippets and usage telemetry to ensure user privacy and security, especially in enterprise environments.

Code Snippets and Prompts

- **Real-time Processing**: Code snippets and prompts are transmitted to GitHub Copilot in real-time to generate suggestions. These snippets are processed only temporarily and are not stored permanently. Once a suggestion is returned, the code snippets are discarded immediately .
- **Data Collection Settings**: Users and organizations have the ability to control whether their code snippets are used for product improvements. In the GitHub Copilot settings, you can deselect the option that allows GitHub to use your code snippets for product enhancements. This setting ensures that your code will not be retained or shared for further processing.

Usage Telemetry

- **User Engagement Data**: This data includes metrics like which suggestions were accepted or dismissed, error rates, and feature usage statistics. This information helps improve Copilot's functionality but does not include specific code content unless the user has enabled telemetry collection.
- **Pseudonymous Identifiers**: The telemetry data collected includes pseudonymous identifiers to help with feature improvement and detecting potential misuse, without directly linking to personal data.

Enterprise Controls

- **Administrative Oversight**: Enterprise administrators can manage these settings at an organizational level to ensure compliance with internal policies and data privacy standards. This includes the ability to disable telemetry collection entirely

Integration with security tools

GitHub Copilot integrates with various security tools to enhance security practices and streamline development workflows. Here’s an overview of how these integrations work and what they offer:

Integration with Security Tools

1. **Microsoft Copilot for Security**:
   - This integration provides a generative AI-powered assistant tailored for security operations. It helps improve the efficiency of security teams by offering insights and assistance at machine speed and scale. This includes leveraging AI to analyze and respond to security threats, providing detailed prompts and suggestions for mitigating risks.
2. **GitHub Advanced Security (GHAS)**:
   - GitHub Copilot integrates with GHAS to help developers identify and fix security issues more efficiently. This includes the automatic detection of vulnerabilities and the provision of secure coding suggestions. The integration aims to unify various security tools and technologies, empowering developers to address security concerns within their workflow seamlessly.
3. **Audit Logs and Telemetry**:
   - GitHub Copilot provides detailed audit logs and telemetry data to help organizations track usage and monitor security-related events. This data can be accessed through REST API endpoints, allowing for comprehensive monitoring and reporting of Copilot activities, which is crucial for maintaining compliance and security standards.

Benefits and Features

- **Enhanced Security**:
  - By integrating AI-driven insights directly into the development process, Copilot helps developers write more secure code from the outset. This proactive approach to security reduces the risk of vulnerabilities and improves the overall security posture of applications.
- **Streamlined Workflows**:
  - The integration with GHAS and other security tools allows developers to receive security alerts and suggestions within their existing workflows, minimizing disruption and increasing productivity.
- **Data Privacy and Control**:
  - Organizations have control over the data processed by GitHub Copilot. They can manage settings to ensure that code snippets and telemetry data are handled in compliance with their internal policies and regulatory requirements