Responsible AI:

Responsible use of GitHub Copilot

1. **Ethical Use**:
    - Avoid harmful or illegal activities.
    - Follow community guidelines and legal standards.
2. **Verification**:
    - Always verify AI-generated code suggestions.
    - Check for security vulnerabilities and maintain code quality.
3. **Accountability**:
    - Users are responsible for reviewing and validating AI suggestions.
    - Ensure compliance with relevant policies and guidelines.
4. **Platform-Specific Guidelines**:
    - IDE: Use ethically within development environments.
    - GitHub.com: Maintain code quality and security on the platform.
    - Mobile: Be cautious and ethical in mobile contexts.
    - CLI: Review commands generated by Copilot before execution.

These points emphasize the importance of ethical behavior, thorough verification, and accountability when using GitHub Copilot across different platforms.

Responsible use of GitHub Copilot pull request summaries

GitHub Copilot Pull Request Summaries help create AI-generated summaries of pull requests, providing an overview and a bulleted list of changes. Key points include:

- **Ethical Use**: Maintain responsibility and review AI-generated content.
- **Verification**: Manually review and verify summaries for accuracy.
- **Limitations**: Acknowledge limitations in scope, processing time, and potential inaccuracies.
- **Use Case**: Enhances productivity by quickly providing context for human reviews.
- **Feedback**: Users can provide feedback to improve the feature.

Responsible use of GitHub Copilot text complition

GitHub Copilot Text Completion assists in writing pull request descriptions by suggesting text as you type. Key points include:

- **Purpose**: Helps provide context quickly to human reviewers, improving productivity.
- **Verification**: Users must review and verify AI-generated suggestions.
- **Limitations**: Acknowledges potential inaccuracies and limitations with large pull requests.
- **Ethical Use**: Encourages responsible usage, avoiding harmful content replication.

GitHub Copilot plans and features:

features:

- **Code completion:** VS, VS Code, Jetbrains IDEs, Azure Data Studio, VIM/Neovim
- **Copilot chat:** available in GitHub mobile, GitHub.com (Enterprise only), IDE’s(VS, VS Code, Jetbrains IDE’s) ,Skills with Enterprise
- Copilot in the CLI: Chatinterface in terminal
- Copilot pull request summeries: Enterprise only! Generated Pull request summeries
- Copilot text completion: Enterprise only! support in PR description generation ie
- Copilot knowledge base: Enterprise only! Context for questions on documentation for Github Copilot (RAG)

admin featues (business and enterprise only):

- policy management: org wide policy administration
- access management: manage access to Copilot for orgs and users
- usage data: view details on if and how Copilot is used
- audit logs: view who has than what
- exclude files: exclude files from Copilot usage

plans:

Subscription plans for GitHub Copilot

GitHub offers multiple subscription options for GitHub Copilot:

- **GitHub Copilot Individual** for individual GitHub customers
- **GitHub Copilot Business** is available for organizations with a GitHub Free or GitHub Team plan, or enterprises on GitHub Enterprise Cloud.
- **GitHub Copilot Enterprise** is available for enterprises on GitHub Enterprise Cloud. Copilot is not currently available for GitHub Enterprise Server.

Comparing Copilot subscriptions

	Copilot Individual	**Copilot Business**	**Copilot Enterprise**
Pricing	$10 USD per month, or$100 USD per year(free for some users)	$19 USD per granted seat per month	$39 USD per granted seat per month
Code completion in IDEs	✓	✓	✓
Copilot Chat in IDEs	✓	✓	✓
Copilot Chat in GitHub Mobile	✓	✓	✓
Copilot in the CLI	✓	✓	✓
Block suggestions matching public code	✓	✓	✓
Exclude specified files from Copilot	✕	✓	✓
Organization-wide policy management	✕	✓	✓
Audit logs	✕	✓	✓
Copilot Chat in GitHub.com	✕	✕	✓
Copilot Chat skills in IDE	✕	✕	✓
Copilot pull request summaries	✕	✕	✓
Copilot knowledge bases	✕	✕	✓

Further feature comparission:

	GitHub Copilot Enterprise	**GitHub Copilot for Business**	**GitHub Copilot for Individual**
Collaborative Chat within pull requests	✓	✕	✕
Lives on Github.com	✓	✓	✓
Plugs right into your editor	✓	✓	✓
Pull request summaries	✓	✕	✕
Copilot docset management	✓	✕	✕

	GitHub Copilot Enterprise	**GitHub Copilot for Business**	**GitHub Copilot for Individual**
Copilot code review	✓	✕	✕
Zero data retention for code snippets and usage telemetry	✓	✓	✓
Organization-wide policy management	✓	✓	✕
Integration with security tools	✓	Limited	✕
Audit logging and reporting	✓	Limited	✕
VPN proxy support via self-signed certificates	✓	✓	✕

How Github Copilot works and handles data:

Overview of GitHub Copilot

GitHub Copilot is an AI-powered coding assistant that integrates into various development environments to help developers write code more efficiently. It provides autocomplete-style code suggestions as you type, leveraging machine learning models to understand and predict the code you need

Key Features

- **Code Completion**: Autocomplete suggestions in supported IDEs (e.g., Visual Studio Code, JetBrains IDEs).
- **Copilot Chat**: A chat interface available in GitHub.com (Enterprise only), GitHub Mobile, and supported IDEs, where you can ask coding-related questions.
- **Command Line Interface (CLI)**: A chat-like interface in the terminal to provide command suggestions or explanations.
- **Pull Request Summaries**: AI-generated summaries of changes in a pull request for a quick overview.
- **Text Completion**: Helps in writing pull request descriptions accurately.
- **Knowledge Bases**: Collections of documentation to provide context in chats with Copilot (Enterprise only

How GitHub Copilot Works

GitHub Copilot uses machine learning models trained on a vast dataset of public code repositories. When you type code in your IDE, Copilot analyzes the context and provides suggestions. The tool continuously learns and improves by using anonymized data from user interactions, ensuring it adapts to diverse coding styles and needs

Data Handling and Privacy

- **Data Collection**: GitHub Copilot collects usage data, such as the code snippets you type, your acceptance of suggestions, and other interactions to improve its models. This data is anonymized to protect user privacy.
- **Content Exclusion**: Users and organizations can configure Copilot to ignore certain files, ensuring sensitive or proprietary code is not used to train models.
- **Usage Data**: Organizations with enterprise subscriptions can review Copilot usage data to manage access and adoption effectively.
- **Audit Logs**: Available for administrators to review actions taken by users, enhancing transparency and security

Prompt Crafting and Prompt Engineering:

General tips:

Start general and get more specific

Top Down principle

Example

Write a function that tells me if a number is prime
The function should take an integer and return true if the integer is prime
The function should give an error if the input is not a positive integer>

Give examples

Add context to your prompts

Example

Write a function that identifies dates in a string and returns them. Dates can have formats like:

- 31/07/2024
- 2024-07-01

Tip

Examples might be as well be unit test

break complex tasks into simpler tasks

kind of obvious should be done in development anyways

Example

- Write a function to generate a 10 by 10 grid of letters.
- Write a function to find all words in a grid of letters, given a list of valid words.
- Write a function to that uses the previous functions to generate a 10 by 10 grid of letters that contains at least 10 words.
- Update the previous function to print the grid of letters and 10 random words from the grid.

avoid ambiguity

be explicit in what you want Copilot to do

Example

What does function X do? rather than *what does this piece of code do?*

indicate relevant code

In your IDE use references to indicate relevant code to Copilot Prompting in the IDE - VS Code

experiment and iterate

Work with previous responses where it makes sense to keep context for Copilot or use */clear* to reset context

keep history relevant

To keep history relevant use threads => start new conversations to keep history of certain topics. Delete requests that are not needed anymore or took a wrong turn

follow good coding practices

Like when developing one your own (clean code):

- consistant coding style
- descriptive names
- comments where needed
- adhere to rules to structure code (modules, components, etc)
- include unit tests

> **Tip**
> use Copilot as support for that as well by asking for tips on coding practices.
> Works also for other areas like asking for possible vulnerabilities

common prerequisites for all IDEs:

- Access to GitHub Copilot
- the IDE (no na ned)
- Sign into GitHub in the IDE

Prompting in the IDE - VS Code:

Prerequesites:

VS Code GitHub Copilot extension
VS Code GitHub Copilot Chat extension

Ways to open chat:

MS resource on that: https://code.visualstudio.com/docs/copilot/overview

- UI: click the chat icon in activity bar
- Inline chat: Control+Command+i (Mac) / Ctrl+Alt+i (Windows/Linux)
- Quick Chat: Shift+Command+i (Mac) / Shift+Ctrl+i (Windows/Linux)
- Smart Actions via context menu (right click)

Use chat:

- type away

- use suggested prompts
- make follow up request to previous prompt

Chat Participants aka Keywords

Built in in VS Code:

- @workspace: Use the code in the current workspace as context for your question
- @vscode: Context is VS Code commands and features
- @terminal: Context is the current terminal shell => help on usage and debugging there
  More are available on GitHub marketplace or VS Code marketplace

  > **Tip**
  > available participants are shown by entering @ in the chat box

Slash commands

Shortcuts for common actions

- /tests: Generates unit tests for selected code
- /fix: propose a fix for problems in selected code
- /explain: explain selected code
- /clear: start new chat
- /new: depends on context new workspace, new file

> **some slash commands like new are context dependent**

Chat in Visual Studio

Prerequesites:

Visual Studio GitHub Copilot extension
Visual Studio GitHub Copilot Chat extension

Ways to open chat:

- VS Menu bar
- inline by right click “Ask Copilot” option

slash commands

- /tests: generate unit tests for selected code
- /fix: propose a fix for issues in selected code
- /explain: explain selected code
- /optimize: analyse and optimize selected code

References:

By default always current opened file
In chat references can be added by # followed by what to reference. Posibilities are:

- a specific file (ie: What does #myclass.cs do?)
- multiple files (ie: Are there duplicated methods in #myclass.cs #myclass1.cs ?)
- reference specific lines (ie: Explain #myclass187.cs: 231-244)
- reference the solution (ie: Is there a datetimeUTC method in #solution?)
- combinations (ie: /explain the deleteItem method in #myApi.cs

Chat in Jetbrains IDEs

List of supported Jetbrains IDEs:

- IntelliJ IDEA (Ultimate, Community, Educational)
- Android Studio
- AppCode
- CLion
- Code With Me Guest
- DataGrip
- DataSpell
- GoLand
- JetBrains Client
- MPS
- PhpStorm
- PyCharm (Professional, Community, Educational)
- Rider
- RubyMine
- RustRover
- WebStorm

All of them have plugin systems and possilities to add a GitHub Copilot Chat extension/addin/plugin possibilites is in general similar to MS IDE integrations. So by default current file is referenced. References can be set. Keywords can be used

Additional info on Chat usage:

Custom extensions:

Custom extensions can be added and adressed in prompts in example: @sentry (this is an existing application monitoring tool) to address an extension listening on the @sentry keyword followed by specific operations.
ie: `@sentry create a GitHub issue for the most recent Sentry issue and assign it to @DEVELOPER`

Benefits of custom extensions:

- Interaction with external tools
- Reduced context switching
- customize Copilot chat experience for better developer workflow

IDE availaibility:

- VS
- VS Code
- Github.com (with Enterprise subscription)

where to get:

GitHub Marketplace

⚠ **this is currently only available as BETA feature for Individuals and organizations or enterprises administrators must grant access to their users. Also for orgs and enterprises the admin has to install them**

Best practices, examples and tips:

Set the stage with a high level goal:

Example:
`/* Create a basic markdown editor in Next.js with the following features: - Use react hooks - Create state for markdown with default text "type markdown here" - A text area where users can write markdown - Show a live preview of the`

```
markdown text as I type - Support for basic markdown syntax like headers, bold,
italics - Use React markdown npm package - The markdown text and resulting HTML
should be saved in the component's state and updated in real time */
```

Make your ask simple and specific

After the main goal go more into the details but be specfic and let Copilot generate steps and not the whole solution at once => better outcome.

Work with examples

When providing details and examples Copilot has an easier time understand intentions.
Example:
Provide examples including values and not only structure for a JSON for which Copilot should generate a DTO.

Experiment with prompts:

use threads to seperate different prompts and do not hesitate to open a new thread to try a different approach

Keep relevant tabs open

Neighboring tabs are used as context for prompts => adds automatically details

Stay smart:

double check provided solutions

Developer use cases for AI:

1. Code Completion

- **Autocomplete Suggestions**: GitHub Copilot provides real-time autocomplete suggestions for various programming languages, such as Python, JavaScript, TypeScript, Ruby, Go, C#, and C++. It assists with writing code faster by predicting the next parts of your code as you type
- **Framework and API Support**: It supports numerous frameworks and APIs, helping generate boilerplate code, API calls, and framework-specific implementations.

2. Copilot Chat

- **Interactive Assistance**: Developers can ask coding-related questions directly in their IDE, such as how to implement a function, fix a bug, or understand a piece of code. This feature is available in GitHub.com, GitHub Mobile, and supported IDEs
- **Documentation and Explanation**: Copilot Chat can explain code snippets, generate comments, and provide documentation for selected lines of code.

3. Command Line Interface (CLI)

- **Command Suggestions**: In the terminal, Copilot can suggest and explain commands, making it easier to work with command-line tools and scripts
- **Interactive Queries**: Users can ask for explanations or alternative ways to execute commands, improving their efficiency in terminal operations.

4. Pull Request Summaries

- **Automated Summaries**: Copilot can generate summaries for pull requests, highlighting changes, impacted files, and key points for reviewers. This feature is particularly useful for streamlining the code review process in large projects

5. Text Completion

- **Description Assistance**: When creating pull requests or writing documentation, Copilot can suggest text completions to help describe changes or functionalities more effectively

6. Knowledge Bases (Enterprise Only)

- **Contextual Information**: Developers can create and manage collections of documentation that Copilot uses as context for providing more accurate and relevant suggestions in chats and coding assistance

7. Prompt Engineering

- **Custom Prompts**: Users can craft specific prompts to guide Copilot in generating the desired output, improving the relevance and accuracy of the suggestions

Testing with Github Copilot:

1. Generating Unit Tests

- **Automatic Test Generation**: Copilot can generate unit tests for your existing code. By typing `/tests` in the chat prompt or using inline commands in your editor, you can ask Copilot to create comprehensive tests that cover various aspects of your functions.
- **Slash Commands**: In Visual Studio Code, you can use slash commands like `/tests` to quickly generate unit tests for selected code. This simplifies the process of writing repetitive test cases and ensures better code coverage

2. Code Suggestions and Completion

- **Functionality Testing**: Copilot provides code suggestions that can be directly used to write test cases. For instance, when you start writing a test function, Copilot can suggest the entire function body, including setup, execution, and assertions.
- **Alternative Suggestions**: When generating test code, Copilot can offer multiple suggestions, allowing you to choose the best fit for your testing needs. This helps in crafting more robust and varied test cases

3. Explaining and Fixing Code

- **Bug Fixing**: Copilot can propose fixes for problems detected in your code. By typing `/fix` followed by the issue description, you can get suggestions on how to resolve specific bugs, including test-related bugs
- **Code Explanation**: You can use Copilot to explain complex parts of your code. This is particularly useful for understanding legacy code or code written by others, ensuring that your tests are comprehensive and accurate

4. Command Line Interface (CLI)

- **Test Commands in CLI**: For those who prefer working in the terminal, Copilot can suggest and explain commands related to testing. By using `gh copilot suggest` or `gh copilot explain`, you can get help with test commands and their explanations directly in the CLI

5. Documentation and Comments

- **In-line Documentation**: Copilot can generate comments and documentation for your test cases. This helps in maintaining clarity and understanding of what each test is intended to verify, making it easier for others to understand and contribute to your test suite

Privacy fundamentals and content exclusions:

Privacy Fundamentals

GitHub Copilot is designed with privacy considerations to protect user data while enhancing productivity. The service collects data to improve its machine learning models, but it also includes measures to respect user privacy and data security:

1. **Data Collection**:
   - GitHub Copilot collects telemetry data, which includes interactions like code completions and modifications, to improve its suggestions. This data is anonymized to protect individual privacy.
   - Copilot also logs how users interact with suggestions, including acceptance, rejection, or modification, to refine its algorithms.
2. **User Control**:
   - Users have control over their data and can manage settings to limit what Copilot accesses and logs. This includes the ability to disable Copilot entirely if desired.
   - Privacy settings are available to customize data sharing preferences, ensuring users can opt out of any data collection they are uncomfortable with.

Content Exclusions

Content exclusions are a crucial feature for organizations using GitHub Copilot, ensuring that sensitive or proprietary code is not used to train Copilot's models. Here's how content exclusions work:

1. **Setting Up Exclusions**:
   - Organizations can specify files or directories that GitHub Copilot should ignore. This is done by configuring content exclusions in the repository or organization settings.
   - Paths can be specified using pattern matching (e.g., `*.secret` to exclude all files with `.secret` extension) to ensure comprehensive exclusion of sensitive data.
2. **Configuration Process**:
   - To set up content exclusions, organization owners need to navigate to the Copilot settings within their GitHub organization settings. Here, they can list repositories and paths to be excluded from Copilot's analysis.
   - The configuration allows using different repository references and pattern matching to tailor the exclusions precisely to organizational needs.
3. **Monitoring and Auditing**:
   - Changes to content exclusions are logged, and organization owners can review these changes. This includes details about who made the changes and when, ensuring transparency and accountability.

- Audit logs help in tracking modifications to exclusion settings, providing an extra layer of security and oversight.

GitHub Copilot Docsets: An Overview

GitHub Copilot Docsets are a feature that provides developers with access to collections of documentation directly integrated with GitHub Copilot. These docsets help developers get the most out of Copilot by offering contextually relevant information and examples as they code. Here are the key aspects of Copilot Docsets:

Features and Functionality

1. **Contextual Documentation**:
    - GitHub Copilot Docsets provide inline documentation and code examples directly within the development environment. This helps developers understand and implement code more efficiently without needing to leave their IDE.
2. **Interactive Assistance**:
    - Copilot can use docsets to provide detailed explanations and suggestions based on the specific documentation collections configured by the user or organization. This is particularly useful for understanding complex codebases or specific libraries and frameworks.
3. **Knowledge Bases for Enterprises**:
    - For enterprise users, GitHub Copilot offers the ability to create and manage collections of documentation known as knowledge bases. These knowledge bases can be tailored to include specific organizational guidelines, code standards, and proprietary documentation, ensuring that Copilot’s suggestions align with company policies and best practices.

Setting Up and Managing Docsets

1. **Configuring Docsets**:
    - Users can configure which documentation sets are used by Copilot through their GitHub settings. This can be done for individual repositories or across an organization to ensure that all relevant documentation is included.
2. **Content Exclusion**:
    - Organizations can exclude certain files or directories from being processed by Copilot, ensuring sensitive or proprietary information is not used to generate suggestions. This is managed through the repository or organization settings, where patterns and specific paths can be defined for exclusion.
3. **Privacy and Security**:

- GitHub Copilot ensures that user data is handled securely. Telemetry data collected is anonymized, and users can manage their privacy settings to control what data is shared. Content exclusions help further protect sensitive information.

4. **Integration with IDEs**:
   - Copilot Docsets integrate seamlessly with various IDEs, such as Visual Studio Code and JetBrains IDEs, providing real-time documentation access and code suggestions. This helps streamline the coding process by reducing context-switching and making relevant information readily available.

GitHub Copilot Enterprise Features

1. **Enhanced Capabilities**:
   - GitHub Copilot Enterprise includes all the features of the individual Copilot, such as code completion, chat-based assistance, command line interface (CLI) support, pull request summaries, and text completion.
2. **Knowledge Bases**:
   - Enterprises can create and manage knowledge bases that include collections of documentation specific to the organization. These knowledge bases can be used as context for Copilot's suggestions, ensuring they are aligned with internal standards and practices.
3. **Content Exclusions**:
   - Organizations can configure content exclusions to prevent specific files or directories from being processed by Copilot. This is crucial for protecting sensitive or proprietary information. Administrators can set these exclusions in the organization settings on GitHub.com
4. **Policy Management**:
   - Enterprises have advanced policy management features. This includes the ability to manage which members of the organization have access to Copilot, what data can be collected, and how it is used. Administrators can review usage data, audit logs, and manage network access to ensure compliance with company policies.
5. **Subscriptions and Access**:
   - GitHub Copilot Enterprise offers different subscription plans tailored for business and enterprise needs. These plans provide enhanced support and features suitable for large teams and complex projects. Users can request access through their organization's GitHub settings.
6. **Advanced Security and Compliance**:
   - Enterprise versions of Copilot include features to help with compliance and security. This includes the ability to audit changes to content exclusion settings and monitor usage across the organization. This ensures that all Copilot interactions comply with the organization's security policies.

GitHub Copilot for Business

GitHub Copilot for Business shares several features with GitHub Copilot Enterprise, tailored to support the needs of business-level users. Here are the key points covered in your inquiries that are relevant to GitHub Copilot for Business:

Key Features and Functionalities

1. **Code Completion and Assistance**:
    - GitHub Copilot for Business provides autocomplete-style code suggestions across various programming languages and frameworks. It helps developers write code faster and with fewer errors by predicting the next lines of code as they type.
2. **Interactive Chat-Based Assistance**:
    - Business users can leverage Copilot Chat to ask questions about their code, get explanations, and receive help with debugging. This feature is available in supported IDEs and helps streamline the development process by providing instant assistance.
3. **Pull Request Summaries**:
    - Copilot for Business can generate summaries for pull requests, highlighting the key changes and potential areas of concern. This helps in quicker and more efficient code reviews.
4. **Command Line Interface (CLI) Support**:
    - Copilot extends its capabilities to the command line, where it can suggest and explain commands. This feature is particularly useful for developers who work extensively in terminal environments.

Management and Administration

1. **Policy Management**:
    - Business users can manage access to Copilot, define usage policies, and control data collection preferences. Administrators have tools to set organizational policies that ensure Copilot is used in a compliant and secure manner.
2. **Content Exclusion**:
    - Organizations can exclude specific files and directories from being processed by Copilot. This feature ensures that sensitive or proprietary information is not used to generate code suggestions, protecting intellectual property and compliance with data security standards.
3. **Subscription and Access**:
    - GitHub Copilot for Business offers subscription plans that provide access to Copilot features for all members of an organization. These plans include options for free trials and are designed to scale with the size of the business.

Privacy and Security

1. **Data Handling and Privacy**:
   - Copilot for Business ensures that data collected is anonymized and used to improve the service. Users can manage their privacy settings to control what data is shared, and administrators can audit these settings for compliance purposes
2. **Audit Logs and Usage Monitoring**:
   - Administrators can access audit logs to review changes to content exclusion settings and monitor overall usage of Copilot within the organization. This transparency helps maintain accountability and security .

Support and Training

1. **Documentation and Knowledge Bases**:
   - Business users have access to extensive documentation and can create knowledge bases that integrate with Copilot. These knowledge bases can include internal documentation and coding standards to ensure consistent and high-quality code across the organization.
2. **Training and Onboarding**:
   - GitHub provides resources and documentation to help businesses onboard new users to Copilot quickly and effectively. This includes quickstart guides, best practices, and example use cases

GitHub Copilot CLI basics:

The GitHub Copilot CLI (Command Line Interface) provides several key commands to enhance your command-line experience using AI assistance directly in the terminal. Here are the main commands available:

1. **`gh copilot suggest [description]`**: This command suggests a command based on the description you provide. For example, you could run `gh copilot suggest "create a new Git branch"` to receive a suggestion for the appropriate command.
2. **`gh copilot explain [command]`**: This command explains the functionality of a given command. For instance, you can use `gh copilot explain "git rebase"` to get a detailed explanation of what that command does.
3. **`gh copilot alias`**: This command generates shell-specific aliases for convenience. These aliases (`ghcs` and `ghce`) can be used to quickly invoke the `suggest` and `explain` functionalities without typing the full command.
4. **`gh copilot config`**: This command allows you to configure options for the Copilot CLI, such as changing the default behavior for command execution confirmations.

These commands help streamline your workflow by providing AI-driven suggestions and explanations directly within your terminal, making it easier to work with complex or unfamiliar commands

GitHub Copilot and Azure DevOps

1. **Licensing Requirements:**

- **GitHub Copilot**: To use GitHub Copilot, users need an active subscription, either through individual, business, or enterprise plans provided by GitHub. This subscription is specifically for Copilot and is independent of any Azure DevOps licensing.
- **Azure DevOps**: Azure DevOps services (like pipelines, repos, and boards) require their own licensing and subscription plans. These are separate from GitHub Copilot. If you're using Azure DevOps, you'll need to ensure you have the necessary licenses for the services you plan to use.

2. **Integration Scenarios:**

- **Using GitHub Copilot with Azure Repos**: While GitHub Copilot is primarily designed to work seamlessly with GitHub repositories, it can also be used with code hosted in Azure Repos, provided that the development environment (like VS Code) where Copilot is enabled is set up to access those Azure Repos. The Copilot subscription still applies in this scenario.
- **Enterprise Integration**: For enterprises that use both GitHub and Azure DevOps, it's possible to manage and integrate workflows across both platforms. However, each platform’s services must be licensed separately. For example, GitHub Actions can be used in conjunction with Azure DevOps pipelines, but this doesn't negate the need for separate licensing.

3. **Cost Implications:**

- **Separate Cost Centers**: The costs associated with GitHub Copilot and Azure DevOps are billed separately. Even if an organization uses both platforms, the subscription for Copilot is not included in Azure DevOps licenses, and vice versa.
- **Bundled Offerings**: Microsoft may offer bundled pricing or discounts for enterprises using multiple Microsoft products, including GitHub and Azure DevOps, but these are usually part of broader enterprise agreements and not specific to Copilot.

4. **Management and Compliance**:

- **Admin Control**: If you're an organization using both Azure DevOps and GitHub (with Copilot), administrators will need to manage licensing and access separately through the respective portals (Azure Portal for Azure DevOps and GitHub Enterprise for Copilot).
- **Compliance**: Both GitHub Copilot and Azure DevOps must adhere to the organization's compliance policies. For example, if certain repositories or codebases need to comply with specific regulatory standards, these must be managed across both platforms.

In summary, while GitHub Copilot and Azure DevOps can be used together in a development workflow, their licensing and costs are managed independently. Organizations need to ensure they have appropriate subscriptions for both tools if they plan to integrate them into their development processes.