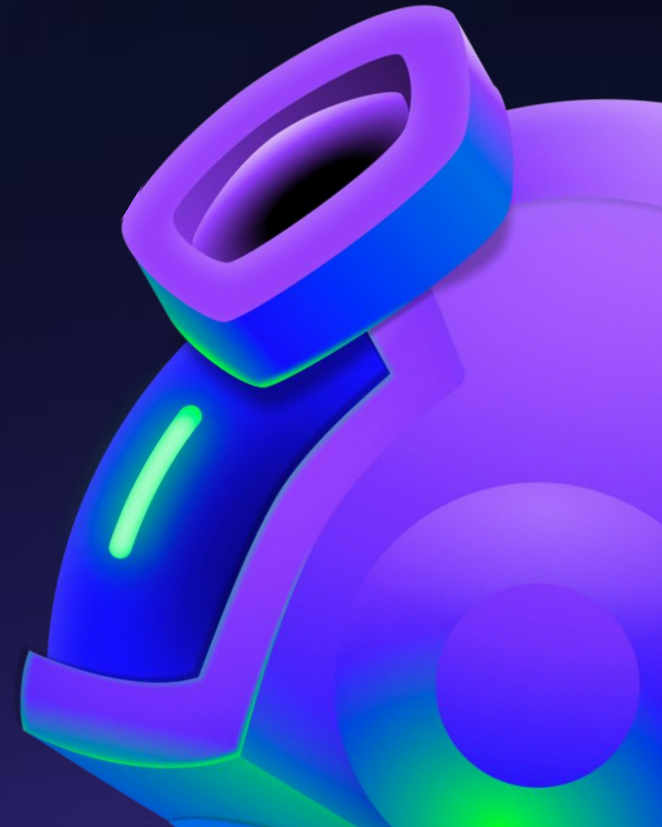




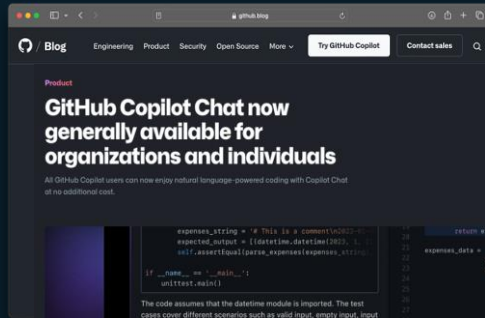
# The World's Most Widely Adopted AI Developer Tool



Dave Burnison  
Senior DevOps Advocate



# GitHub Copilot Journey



Jun '22

Copilot Individual GA



Dec '22

Copilot Business GA  
sales-serve

Feb '23

Copilot Business GA

Dec '23

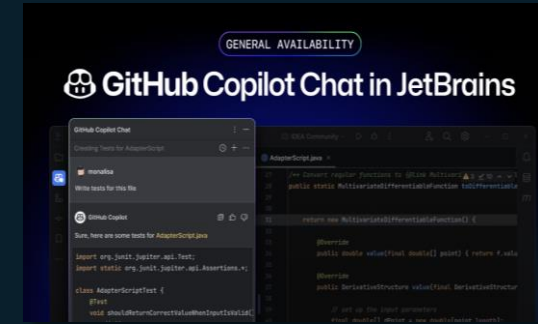
Copilot Chat GA  
Visual Studio & VS Code

Feb '24

Copilot Enterprise GA

Mar '24

Copilot Chat GA  
JetBrains





# GitHub Copilot

Your AI pair programmer

**Offers code suggestions,  
contextualized to your project**

Convert comments to code

Autofill for repetitive code

Autosuggest tests

Show alternatives



```
sentiment.ts  write_sql.go  parse_expense

1  #!/usr/bin/env ts-node
2
3  import { fetch } from "fetch-h2";
4
5  // Determine whether the sentiment of
6  // Use a web service
7  async function isPositive(text: string) {
8      const response = await fetch(`http://
9          method: "POST",
10         body: `text=${text}`,
11         headers: {
12             "Content-Type": "application/x-w
13         },
14     });
15     const json = await response.json();
16     return json.label === "pos";
17 }
```

Copilot

shelterapp > models.py

```
1 from django.db import models
2 from django.core.validators import RegexValidator, MinValueValidator, MaxValueValidator
3 from django.urls import reverse
4
5 # Create your models here.
6
```

# Video: GitHub Copilot Demo

Christopher Harrison, @geektrainer

<https://youtu.be/HDsJQVa0R94>



shelterapp &gt; models.py &gt; Talk

```
1  from django.db import models
2  from django.core.validators import RegexValidator, MinValueValidator, MaxValueValidator
3  from django.urls import reverse
4
5  # Create your models here.
6  # Create a model for Speaker with a name, bio and linkedin URL
7  class Speaker(models.Model):
8      name = models.CharField(max_length=100)
9      bio = models.TextField()
10     linkedin_url = models.URLField(blank=True)
11
12     def __str__(self):
13         return self.name
14
15     # Create a model for Talk with a title, description, speaker and duration
16     class Talk(models.Model):
17         title = models.CharField(max_length=100)
18         description = models.TextField()
19         # Add speaker, but do not allow deletion if talks exist
20         speaker = models.ForeignKey(Speaker, on_delete=models.PROTECT)
21         duration = models.IntegerField(validators=[MinValueValidator(1), MaxValueValidator(120)])
22
23         # Add talk code with format of 3 letters, dash and 3 numbers, add an error message, and make it unique
24         talk_code = models.CharField(max_length=7, validators=[RegexValidator(regex='^[A-Z]{3}-[0-9]{3}$', message='Talk code
25
26     def __str__(self):
27         return self.title
```



# 15 Minutes to Merge

## Help GitHub Copilot help you with prompt crafting



The screenshot shows a code editor with three tabs: `admin.py`, `models.py`, and `views.py`. The `models.py` tab is active, showing the following code:

```
agenda > models.py > ...
1 from django.db import models
2 # Import regex validator
3 from django.core.validators import RegexValidator
4
5 # Import pet user
6 from conference_auth.models import ConferenceAuth
7
8 # import lazy reverse
9 from django.urls import reverse_lazy
10
11 # Create your models here.
12 # Creating a site for a conference
13 class Site(models.Model):
14     name = models.CharField(max_length=100)
15     url = models.CharField(max_length=100)
16     status_choices = (
17         ('submitted', 'Submitted'),
18         ('approved', 'Approved'),
19         ('rejected', 'Rejected'),
20     )
21     status = models.CharField(max_length=20, choices=status_choices)
22     created_at = models.DateTimeField(auto_now_add=True)
23     updated_at = models.DateTimeField(auto_now=True)
24     def __str__(self):
25         return self.name
26     class Meta:
27         verbose_name = 'Site'
28         verbose_name_plural = 'Sites'
29         ordering = ('created_at',)
```

On the right side, there is a chat window titled "CHAT: GITHUB COPIL...". It shows a conversation with "GeekTrainer" and "GitHub Copilot".

**GeekTrainer:** How can I ensure only speakers can edit their own talks?

**GitHub Copilot:** Used 1 reference > To ensure that only speakers can edit their own talks, you can use the `is_owner` method of the `Talk` model. For example, you can use the following code snippet:

```
if talk.is_owner(user):
    # Only speakers can edit their own talks
```

Below the chat window, there is a text input field with the placeholder text "Ask Copilot or type / for commands".

<https://www.youtube.com/watch?v=GPLUGJsVx0s>

<https://docs.github.com/en/copilot/using-github-copilot/prompt-engineering-for-github-copilot>

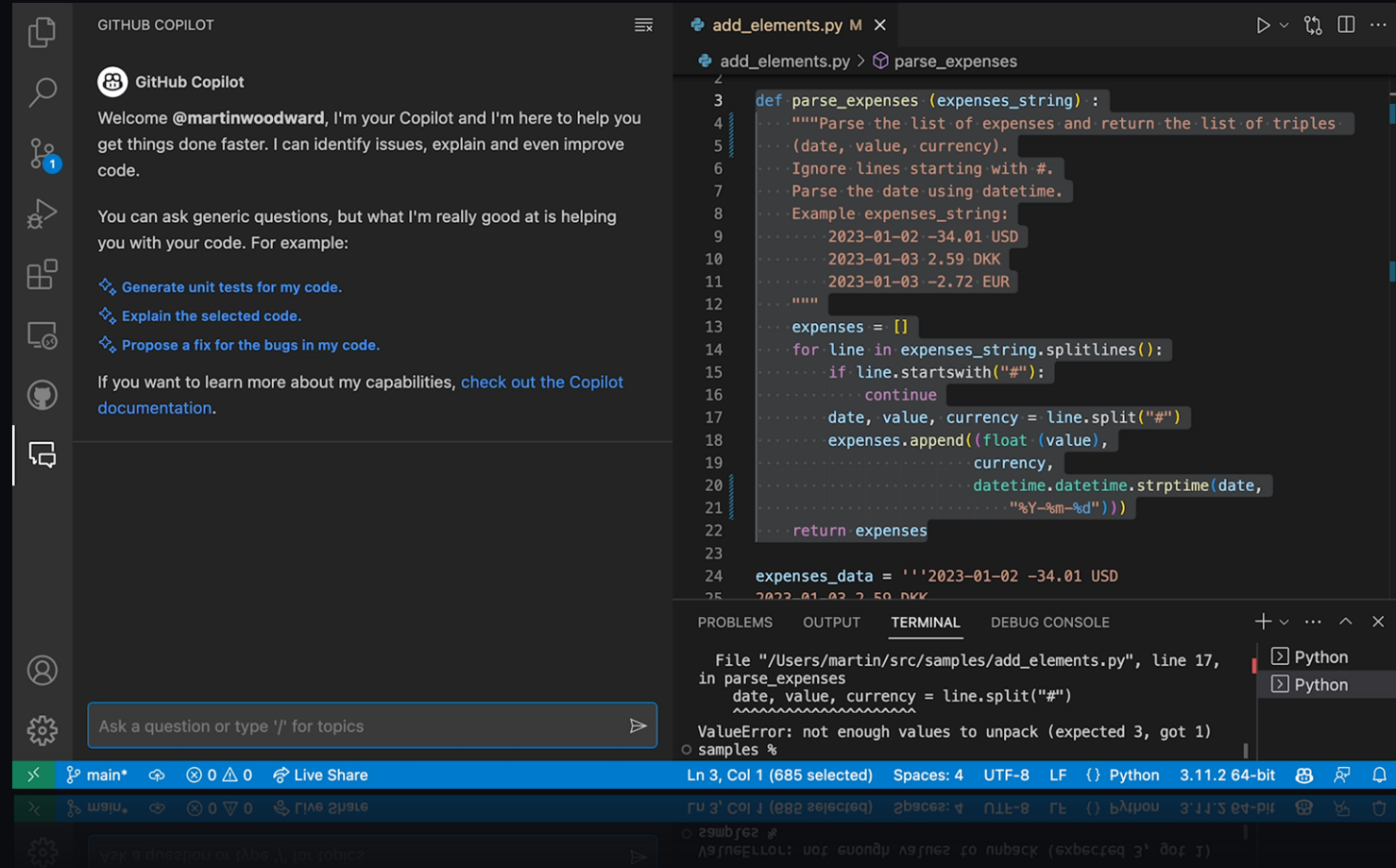
<https://learn.microsoft.com/en-us/training/modules/introduction-prompt-engineering-with-github-copilot>

# GitHub Copilot Chat

Chat experience in the IDE

Interactive experience in editor

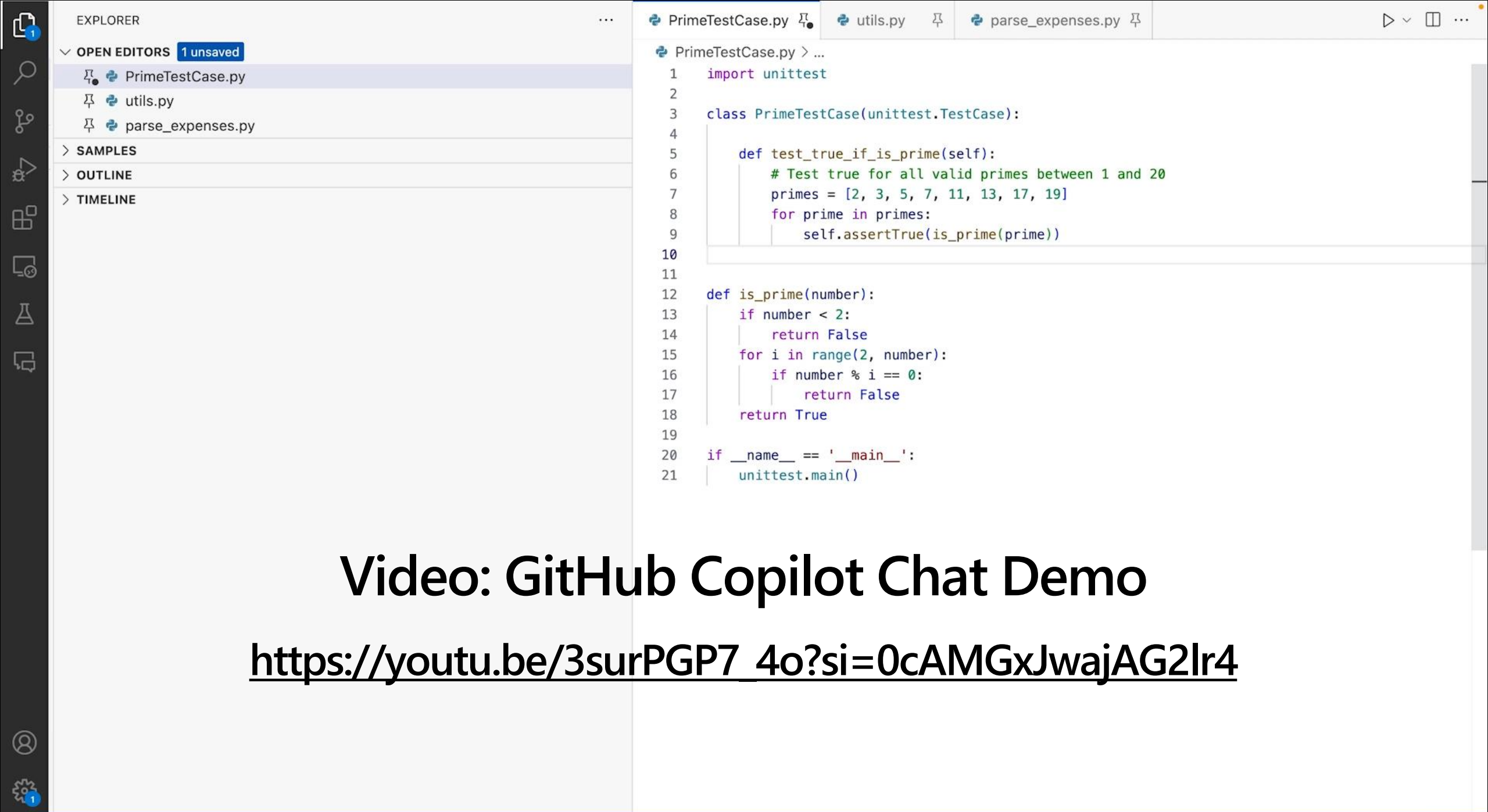
- Get an overview of the solution
- Explain selected code
- Generate unit tests
- Propose a fix for bug
- Make code robust / secure
- Ask generic software development questions



Supports VS Code, Visual Studio and JetBrains.







# Video: GitHub Copilot Chat Demo

[https://youtu.be/3surPGP7\\_4o?si=0cAMGxJwajAG2lr4](https://youtu.be/3surPGP7_4o?si=0cAMGxJwajAG2lr4)

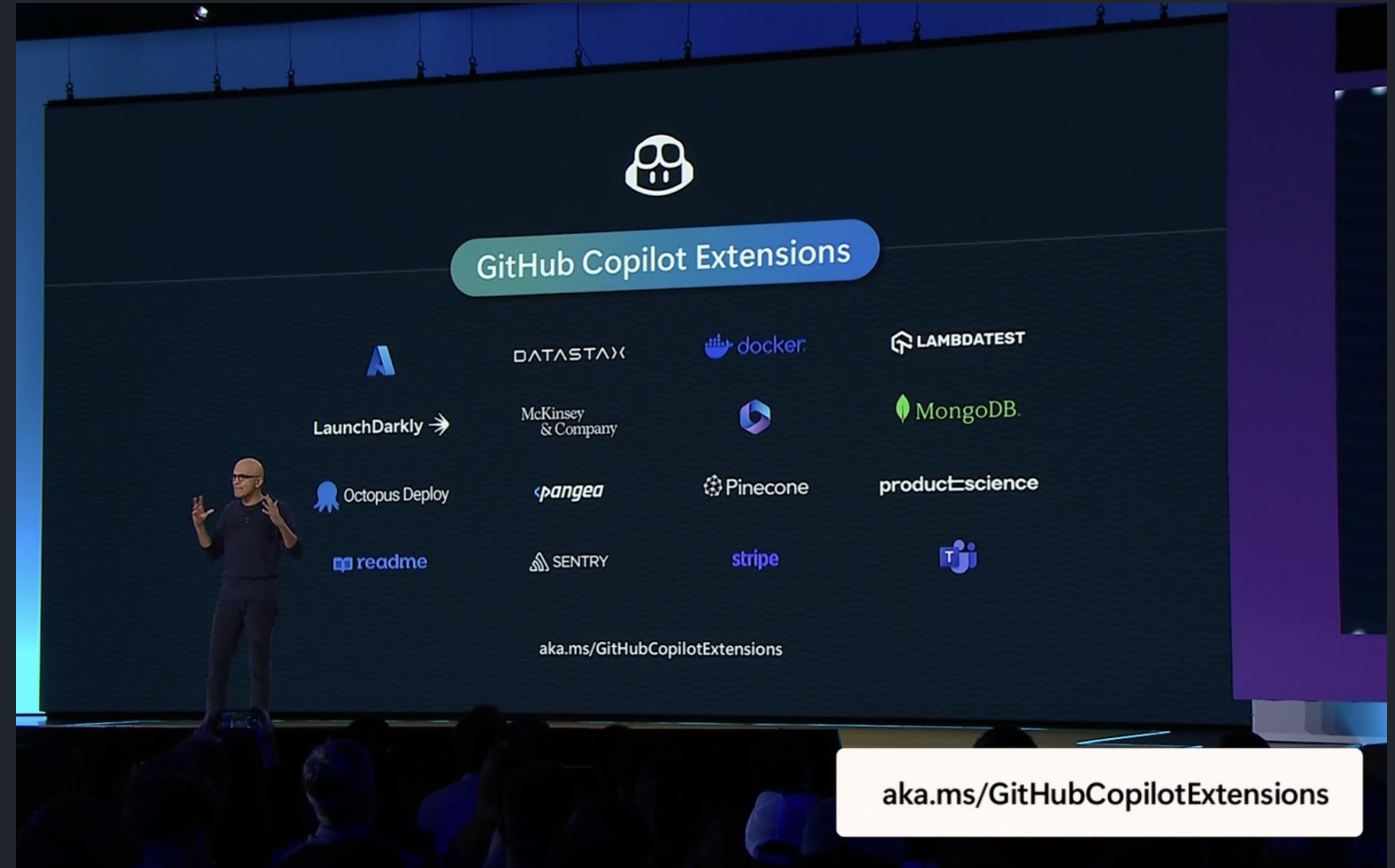




# GitHub Copilot Extensions

Microsoft Build Keynote Address

- GitHub Copilot Extensions bring the world's knowledge into the most widely adopted AI developer tool.
- Through a growing partner ecosystem, Copilot Extensions enable developers to build and deploy to the cloud in their natural language with their preferred tools and services, all without leaving the IDE or GitHub.com.
- Developers can stay in the flow longer, uplevel their skills, and innovate faster.



<http://aka.ms/GitHubCopilotExtensions>, [GitHub Copilot Extension Documentation](https://gh.io/copilot-extension-documentation)  
& <https://gh.io/copilot-partners>

Note: Not associated with any GitHub extension.

# Introducing Copilot Enterprise

- Expands Copilot's AI capabilities from the coding environment to the GitHub Platform.
- Uses the full context of an enterprise's codebase to personalize Copilot across the developer workflow.

## Copilot Enterprise

Copilot personalized for your organization throughout the software development lifecycle

**\$39** per user / month

- ← Everything in Copilot Business, plus...
- ✓ Chat personalized to your codebase
- ✓ Documentation search and summaries
- ✓ Pull request summaries
- ✓ Code review skills
- ✓ Fine-tuned models

\*Requires GitHub Enterprise Cloud

<https://github.blog/2024-02-27-github-copilot-enterprise-is-now-generally-available>

GENERAL AVAILABILITY



# Copilot Enterprise



**Videos: GitHub Copilot Enterprise Demos**  
[https://youtu.be/vUX5u\\_4B2AM?si=sT04fxvXtEd3TN0t](https://youtu.be/vUX5u_4B2AM?si=sT04fxvXtEd3TN0t)

# Copilot Chat in the Repository

- Analyze and explain code and other repository contents
- Refine by selecting which repo files to focus on

The screenshot displays the GitHub Copilot Chat interface. On the left, a code editor shows a Ruby method `def detect(blob, allow_empty: false)` from the file `linguist.rb`. The method's purpose is to detect the language of a given blob of text. On the right, the Copilot chat window shows a user query: "What does this method do?". The Copilot response explains that the `detect` method is used to detect the language of a given blob of text, returning a `Language` object or `nil` if the blob is binary, empty, or if no language is detected. The response also mentions an optional `allow_empty` argument. Below the chat window, there is a search bar with the text "linguist.rb:20-49" and a "Ask Copilot" button.

```
# Returns Language or nil.
def detect(blob, allow_empty: false)
  # Bail early if the blob is binary or empty.
  return nil if blob.likely_binary? || blob.binary? || (!allow_empty && blob.empty?)

  Linguist.instrument("linguist.detection", :blob => blob) do
    # Call each strategy until one candidate is returned.
    languages = []
    returning_strategy = nil

    STRATEGIES.each do |strategy|
      returning_strategy = strategy
      candidates = Linguist.instrument("linguist.strategy", :blob => blob, :strategy => strategy) do
        strategy.call(blob, languages)
      end
      if candidates.size == 1
        languages = candidates
        break
      elsif candidates.size > 1
        # More than one candidate was found, pass them to the next strategy.
        languages = candidates
      else
        # No candidates, try the next strategy
      end
    end

    Linguist.instrument("linguist.detected", :blob => blob, :strategy => returning_strategy,
      languages.first
    end
  end
end
```

arfonsmith now  
What does this method do?

1 Reference ▾  
<> linguist.rb:20-49 ...

Copilot now  
This is the `detect` method in the Ruby file `linguist.rb`.  
  
The method is used to detect the language of the given blob of text. The `blob` object is an instance of a class that includes the `Linguist::BlobHelper` interface. It returns a `Language` object or `nil` if the blob is binary, empty, or if no language is detected.  
  
The method works by first checking whether the blob is binary or empty. Next, it runs a set of language detection strategies with the given blob and the candidate languages returned by the previous strategy, until one candidate is returned. Finally, it returns the first language candidate, or `nil` if none are detected.  
  
This method has an optional argument called `allow_empty` which defaults to false. If it's set to true it will allow detection of languages from empty BLOBs.

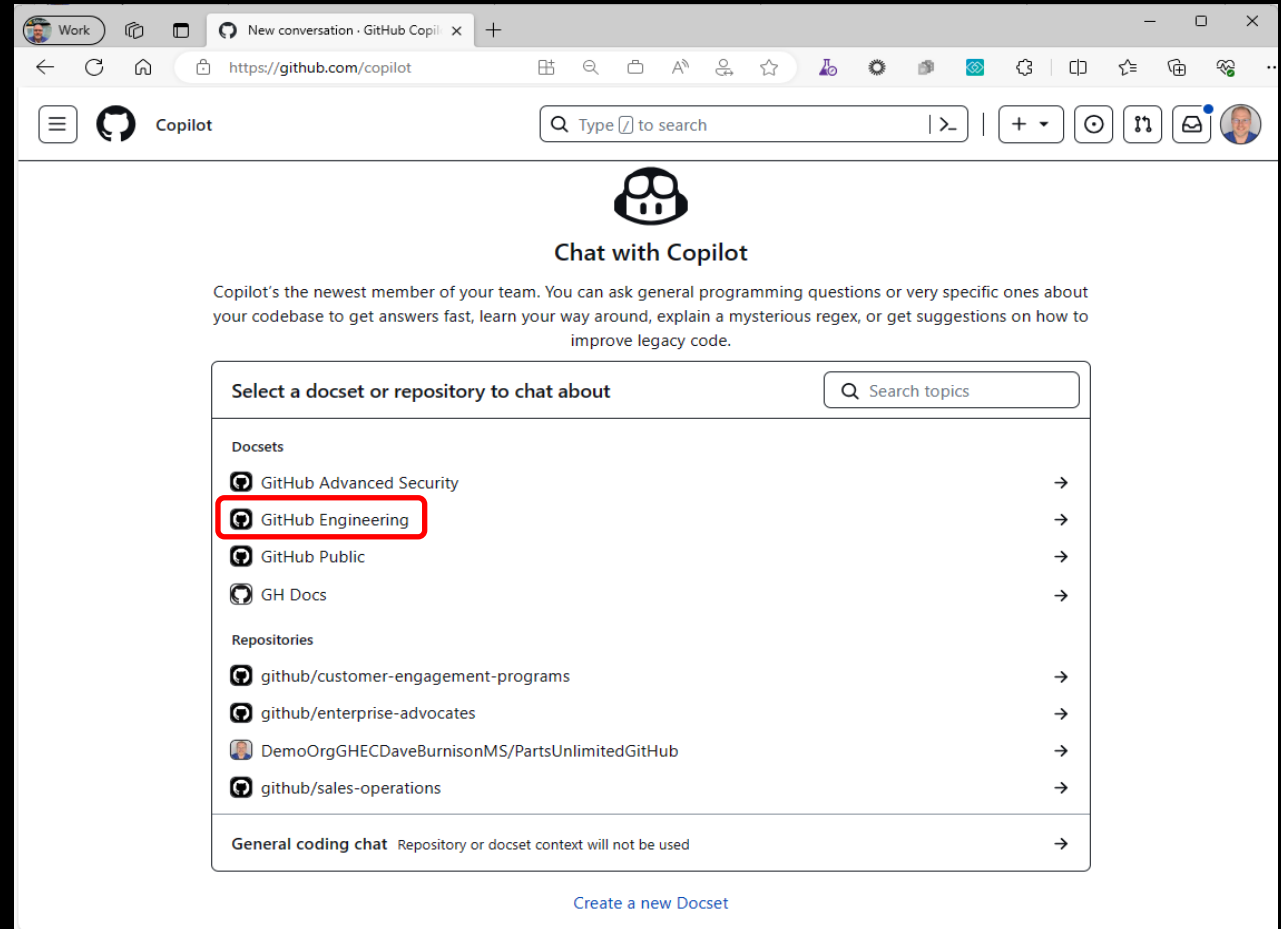
Ask Copilot  
linguist.rb:20-49 X





# Search effortlessly with Copilot Knowledge Bases

- Conversational documentation search
- Customized to your internal documentation
- Summaries and references to source documents



Available in GitHub Copilot Enterprise

Demo: <https://youtu.be/ZDbk5M4hbEI?si=O0T7db3FfxIwlhPW>



# Search effortlessly with Copilot Knowledge Bases

- Conversational documentation search
- Customized to your internal documentation
- Summaries and references to source documents



Available in GitHub Copilot Enterprise

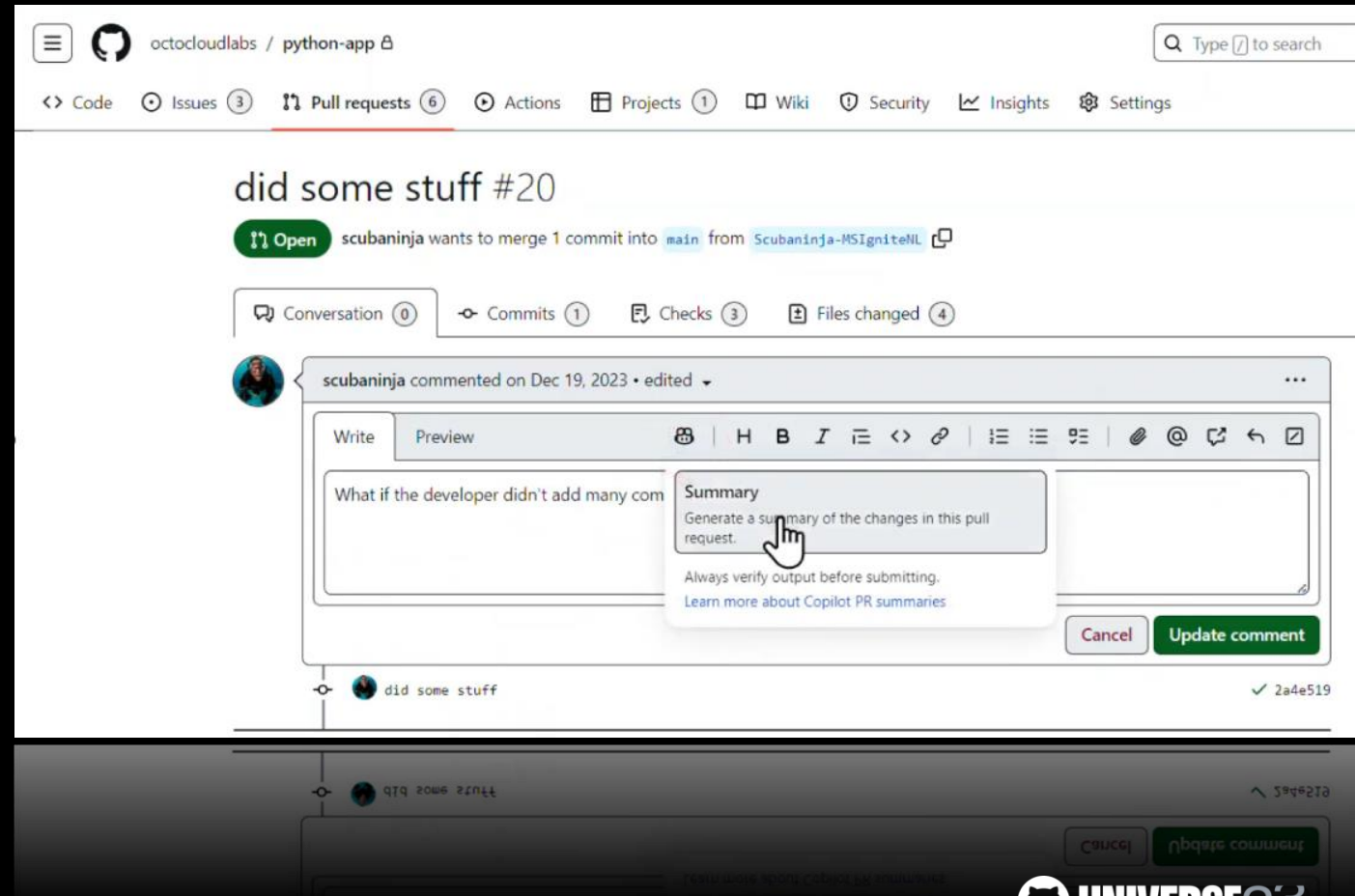
Demo: <https://youtu.be/ZDbk5M4hbEI?si=O0T7db3FfxIwlhPW>

The screenshot shows the GitHub Copilot chat interface. At the top, there's a header with the Copilot logo and navigation icons. Below that, a search bar contains the text "Getting Started with Feature Flags in GitHub" and "GitHub Engineering". A "Beta" badge and a "Give feedback" link are also visible. The main chat area shows a conversation between a user (DaveBurnisonMS) and Copilot. The user's message is "How do I get started using feature flags here at GitHub?". Copilot's response is a list of 8 steps to get started using feature flags at GitHub. The steps are: 1. Think of a name for your flag following the rough pattern `<area_of_code>_<action>_<features>`. 2. Gate the old and new functionality behind the flag using the Flipper library with a construct like `if Flipper.enabled?(:flag_name, app)` where `app` is a specific `Deployment::Application`. 3. Write tests to validate both branches. 4. Get a review and deploy the PR with your changes as normal. 5. Create the flag in the Flipper UI. 6. Post in the #deploys-team that you are enabling a flag so the team can correlate any issues or errors with the change. 7. Flags should be short-lived in most cases. Once you've tested your change and the flag has been fully enabled long enough to validate it, submit a PR removing the flag and old codepath then delete it in the Flipper UI. 8. Flags that are meant to be around for more than a month should be prepended with the `11_` to denote long lived feature flag. Below the steps, Copilot reminds the user to test their feature flag with it both enabled and disabled to make sure both sides of the flag operate as expected with your changes. It also mentions that you can enable the feature flag for a limited testing audience to build confidence, then slowly rollout your flag across environments by slowly incrementing access to wider audiences. This can be done with percentage of users or percentage of calls increases (e.g. 2%, 10%, 30%, 50%, etc.). If you detect any problems you can immediately reduce your rollout percentage or fully disable the feature flag. At the bottom of the chat area, there's a section titled "Search results from GitHub Engineering" which lists 6 search results: 1 - feature\_flags.md:1-99, 2 - index.md:30-42, 3 - feature-flag-rollout.md:11-20, 4 - overview.md:132-151, 5 - feature\_management.md:89-90, and 6 - e2e\_tests.md:198-290. The "Ask Copilot" button is at the bottom right of the chat area.



# Pull request summaries

- Suggest natural-language descriptions within PRs based on the code changes
- Automated code reviews
- Explanations on changes and comments



# Works with any Language

The screenshot displays the GitHub web interface for a repository named "KnowledgePending / COBOL-Fibonacci-Sequence". The left sidebar shows the file structure with "fib.cbl" selected. The main area shows the code for "fib.cbl", which is a COBOL program for calculating the Fibonacci sequence. A red box highlights the file path "COBOL-Fibonacci-Sequence / fib.cbl".

On the right, a "New conversation" chat window is open. A user named "DaveBurnisonMS" has asked, "What does this code do". The Copilot response is as follows:

**Copilot** Responding...

This COBOL program calculates and displays the Fibonacci sequence up to a certain number given by the user. Here's a breakdown:

1. The program first prompts the user to enter a number `N`, which is the position in the Fibonacci sequence the program should calculate up to.
2. Depending on the user's input, the program then goes through several cases:
  - If `N` is 0, it displays 0
  - If `N` is 1, it displays 0 and 1.
  - If `N` is 2, it displays 0, 1 and 1.





Technical Preview



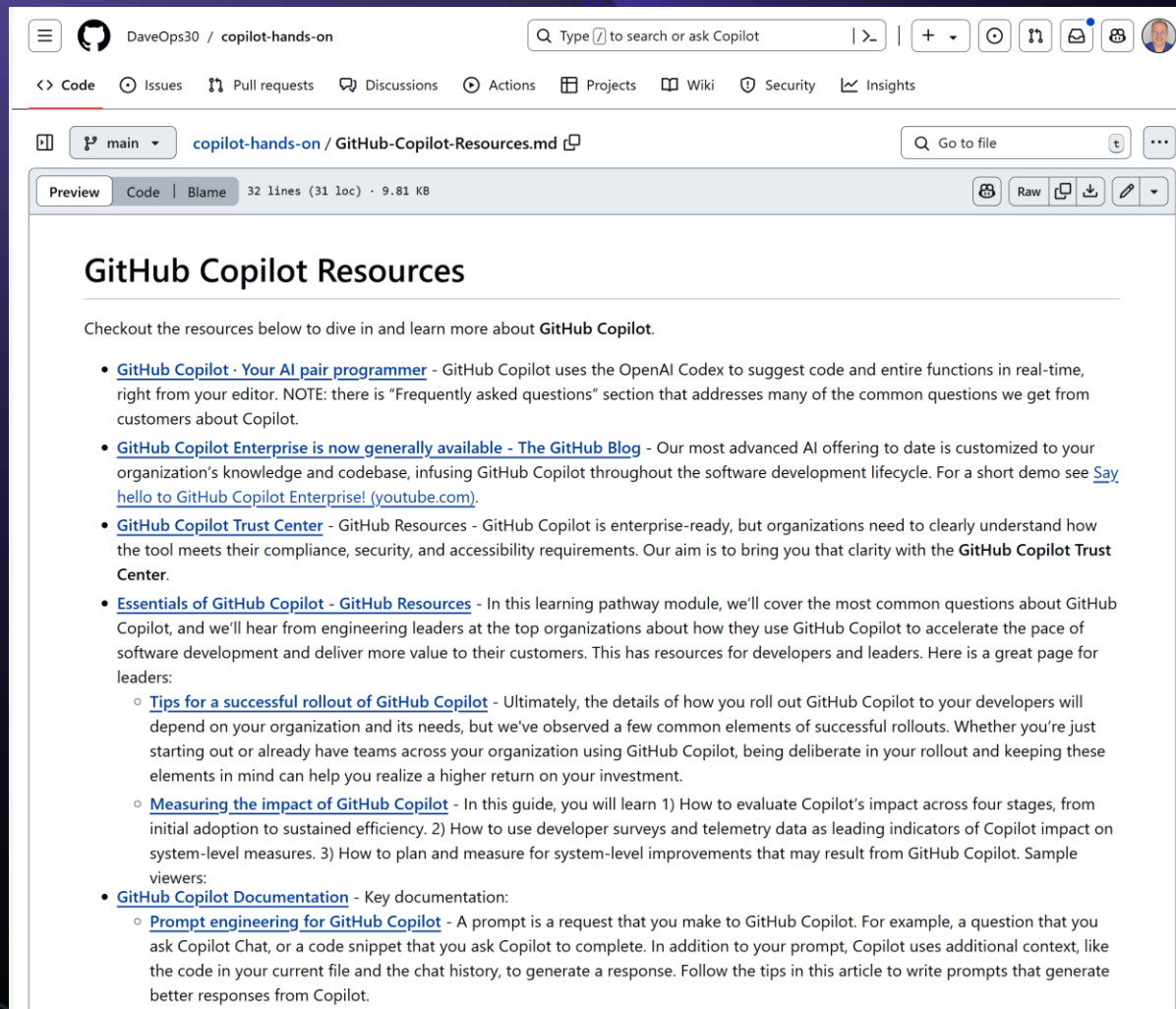
# GitHub Copilot Workspace

Blog Post: <https://github.blog/2024-04-29-github-copilot-workspace>

Demos: <https://youtu.be/L5Xny6yehUg?si=uglP3sIS38bw6lDT> &  
<https://www.youtube.com/watch?v=thyJ9LeBfMI>



# Resources



The screenshot shows a GitHub repository page for 'DaveOps30 / copilot-hands-on'. The file 'GitHub-Copilot-Resources.md' is selected, showing a preview of the content. The page title is 'GitHub Copilot Resources'. The content includes a paragraph and a bulleted list of resources.

Checkout the resources below to dive in and learn more about **GitHub Copilot**.

- [GitHub Copilot - Your AI pair programmer](#) - GitHub Copilot uses the OpenAI Codex to suggest code and entire functions in real-time, right from your editor. NOTE: there is "Frequently asked questions" section that addresses many of the common questions we get from customers about Copilot.
- [GitHub Copilot Enterprise is now generally available - The GitHub Blog](#) - Our most advanced AI offering to date is customized to your organization's knowledge and codebase, infusing GitHub Copilot throughout the software development lifecycle. For a short demo see [Say hello to GitHub Copilot Enterprise! \(youtube.com\)](#).
- [GitHub Copilot Trust Center](#) - GitHub Resources - GitHub Copilot is enterprise-ready, but organizations need to clearly understand how the tool meets their compliance, security, and accessibility requirements. Our aim is to bring you that clarity with the **GitHub Copilot Trust Center**.
- [Essentials of GitHub Copilot - GitHub Resources](#) - In this learning pathway module, we'll cover the most common questions about GitHub Copilot, and we'll hear from engineering leaders at the top organizations about how they use GitHub Copilot to accelerate the pace of software development and deliver more value to their customers. This has resources for developers and leaders. Here is a great page for leaders:
  - [Tips for a successful rollout of GitHub Copilot](#) - Ultimately, the details of how you roll out GitHub Copilot to your developers will depend on your organization and its needs, but we've observed a few common elements of successful rollouts. Whether you're just starting out or already have teams across your organization using GitHub Copilot, being deliberate in your rollout and keeping these elements in mind can help you realize a higher return on your investment.
  - [Measuring the impact of GitHub Copilot](#) - In this guide, you will learn 1) How to evaluate Copilot's impact across four stages, from initial adoption to sustained efficiency. 2) How to use developer surveys and telemetry data as leading indicators of Copilot impact on system-level measures. 3) How to plan and measure for system-level improvements that may result from GitHub Copilot. Sample viewers:
- [GitHub Copilot Documentation](#) - Key documentation:
  - [Prompt engineering for GitHub Copilot](#) - A prompt is a request that you make to GitHub Copilot. For example, a question that you ask Copilot Chat, or a code snippet that you ask Copilot to complete. In addition to your prompt, Copilot uses additional context, like the code in your current file and the chat history, to generate a response. Follow the tips in this article to write prompts that generate better responses from Copilot.

<https://gh.io/ghdevrel-copilot-resources>

# Thank You!

## Have a Question?

Stop by the **GitHub** table  
in the **Microsoft** booth.

