[Adding repository custom instructions for GitHub Copilot - GitHub Docs](https://docs.github.com/en/copilot/how-tos/configure-custom-instructions/add-repository-instructions)

**Adding repository custom instructions for GitHub Copilot**

Create a file in a repository that gives Copilot additional context for the work it does in that repository.

**Tool navigation**

* [Eclipse](https://docs.github.com/en/copilot/how-tos/configure-custom-instructions/add-repository-instructions?tool=eclipse)
* [JetBrains IDEs](https://docs.github.com/en/copilot/how-tos/configure-custom-instructions/add-repository-instructions?tool=jetbrains)
* [Visual Studio](https://docs.github.com/en/copilot/how-tos/configure-custom-instructions/add-repository-instructions?tool=visualstudio)
* [Visual Studio Code](https://docs.github.com/en/copilot/how-tos/configure-custom-instructions/add-repository-instructions?tool=vscode)
* [Web browser](https://docs.github.com/en/copilot/how-tos/configure-custom-instructions/add-repository-instructions?tool=webui)
* [Xcode](https://docs.github.com/en/copilot/how-tos/configure-custom-instructions/add-repository-instructions?tool=xcode)

**In this article**

[About repository custom instructions for Copilot](https://docs.github.com/en/copilot/how-tos/configure-custom-instructions/add-repository-instructions" \l "about-repository-custom-instructions-for-copilot)

[Prerequisites for repository custom instructions](https://docs.github.com/en/copilot/how-tos/configure-custom-instructions/add-repository-instructions" \l "prerequisites-for-repository-custom-instructions)

[Creating a repository custom instructions file](https://docs.github.com/en/copilot/how-tos/configure-custom-instructions/add-repository-instructions" \l "creating-a-repository-custom-instructions-file)

[Writing effective repository custom instructions](https://docs.github.com/en/copilot/how-tos/configure-custom-instructions/add-repository-instructions" \l "writing-effective-repository-custom-instructions)

[Repository custom instructions in use](https://docs.github.com/en/copilot/how-tos/configure-custom-instructions/add-repository-instructions" \l "repository-custom-instructions-in-use)

[Enabling or disabling repository custom instructions](https://docs.github.com/en/copilot/how-tos/configure-custom-instructions/add-repository-instructions" \l "enabling-or-disabling-repository-custom-instructions)

This version of this article is for using repository custom instructions on the GitHub website. Click the tabs above for information on using custom instructions in other environments.

[**About repository custom instructions for Copilot**](https://docs.github.com/en/copilot/how-tos/configure-custom-instructions/add-repository-instructions#about-repository-custom-instructions-for-copilot)

Repository custom instructions let you provide Copilot with repository-specific guidance and preferences.

Repository custom instructions are currently supported for:

* **Copilot Chat** in VS Code
* **Copilot coding agent**
* **Copilot Chat** in Visual Studio, JetBrains IDEs, Xcode, Eclipse, and on the GitHub website (copilot-instructions.md file only)
* **Copilot code review** (copilot-instructions.md file only)

[**Prerequisites for repository custom instructions**](https://docs.github.com/en/copilot/how-tos/configure-custom-instructions/add-repository-instructions#prerequisites-for-repository-custom-instructions)

* You must have a custom instructions file (see the instructions below).
* Your personal choice of whether to use custom instructions must be set to enabled. This is enabled by default. See [Enabling or disabling repository custom instructions](https://docs.github.com/en/copilot/how-tos/configure-custom-instructions/add-repository-instructions#enabling-or-disabling-repository-custom-instructions) later in this article.

[**Creating a repository custom instructions file**](https://docs.github.com/en/copilot/how-tos/configure-custom-instructions/add-repository-instructions#creating-a-repository-custom-instructions-file)

**Copilot Chat** on the GitHub website, **Copilot coding agent** and **Copilot code review** support a single .github/copilot-instructions.md custom instructions file stored in the repository.

In addition, **Copilot coding agent** supports one or more .instructions.md files stored within .github/instructions in the repository. Each file can specify applyTo frontmatter to define what files or directories its instructions apply to.

[**Using a single .github/copilot-instructions.md file**](https://docs.github.com/en/copilot/how-tos/configure-custom-instructions/add-repository-instructions#using-a-single-githubcopilot-instructionsmd-file-1)

You can ask Copilot coding agent to generate a .github/copilot-instructions.md file, or you can write your own instructions file.

[**Asking Copilot coding agent to generate a .github/copilot-instructions.md file**](https://docs.github.com/en/copilot/how-tos/configure-custom-instructions/add-repository-instructions#asking-copilot-coding-agent-to-generate-a-githubcopilot-instructionsmd-file)

Note

Copilot coding agent is in public preview and subject to change. During the preview, use of the feature is subject to [GitHub Pre-release License Terms](https://docs.github.com/en/site-policy/github-terms/github-pre-release-license-terms).

1. Navigate to the Agents page at [github.com/copilot/agents](https://github.com/copilot/agents).

You can also reach this page by clicking the  button next to the search bar on any page on GitHub, then selecting **Agents** from the sidebar.

1. Using the dropdown menu in the prompt field, select the repository you want Copilot to generate custom instructions for.
2. Copy the following prompt, customizing it if needed:

Markdown

Your task is to "onboard" this repository to Copilot coding agent by adding a .github/copilot-instructions.md file in the repository that contains information describing how a coding agent seeing it for the first time can work most efficiently.

You will do this task only one time per repository and doing a good job can SIGNIFICANTLY improve the quality of the agent's work, so take your time, think carefully, and search thoroughly before writing the instructions.

<Goals>

- Reduce the likelihood of a coding agent pull request getting rejected by the user due to

generating code that fails the continuous integration build, fails a validation pipeline, or

having misbehavior.

- Minimize bash command and build failures.

- Allow the agent to complete its task more quickly by minimizing the need for exploration using grep, find, str*\_replace\_*editor, and code search tools.

</Goals>

<Limitations>

- Instructions must be no longer than 2 pages.

- Instructions must not be task specific.

</Limitations>

<WhatToAdd>

Add the following high level details about the codebase to reduce the amount of searching the agent has to do to understand the codebase each time:

<HighLevelDetails>

- A summary of what the repository does.

- High level repository information, such as the size of the repo, the type of the project, the languages, frameworks, or target runtimes in use.

</HighLevelDetails>

Add information about how to build and validate changes so the agent does not need to search and find it each time.

<BuildInstructions>

- For each of bootstrap, build, test, run, lint, and any other scripted step, document the sequence of steps to take to run it successfully as well as the versions of any runtime or build tools used.

- Each command should be validated by running it to ensure that it works correctly as well as any preconditions and postconditions.

- Try cleaning the repo and environment and running commands in different orders and document errors and and misbehavior observed as well as any steps used to mitigate the problem.

- Run the tests and document the order of steps required to run the tests.

- Make a change to the codebase. Document any unexpected build issues as well as the workarounds.

- Document environment setup steps that seem optional but that you have validated are actually required.

- Document the time required for commands that failed due to timing out.

- When you find a sequence of commands that work for a particular purpose, document them in detail.

- Use language to indicate when something should always be done. For example: "always run npm install before building".

- Record any validation steps from documentation.

</BuildInstructions>

List key facts about the layout and architecture of the codebase to help the agent find where to make changes with minimal searching.

<ProjectLayout>

- A description of the major architectural elements of the project, including the relative paths to the main project files, the location

of configuration files for linting, compilation, testing, and preferences.

- A description of the checks run prior to check in, including any GitHub workflows, continuous integration builds, or other validation pipelines.

- Document the steps so that the agent can replicate these itself.

- Any explicit validation steps that the agent can consider to have further confidence in its changes.

- Dependencies that aren't obvious from the layout or file structure.

- Finally, fill in any remaining space with detailed lists of the following, in order of priority: the list of files in the repo root, the

contents of the README, the contents of any key source files, the list of files in the next level down of directories, giving priority to the more structurally important and snippets of code from key source files, such as the one containing the main method.

</ProjectLayout>

</WhatToAdd>

<StepsToFollow>

- Perform a comprehensive inventory of the codebase. Search for and view:

- README.md, CONTRIBUTING.md, and all other documentation files.

- Search the codebase for build steps and indications of workarounds like 'HACK', 'TODO', etc.

- All scripts, particularly those pertaining to build and repo or environment setup.

- All build and actions pipelines.

- All project files.

- All configuration and linting files.

- For each file:

- think: are the contents or the existence of the file information that the coding agent will need to implement, build, test, validate, or demo a code change?

- If yes:

- Document the command or information in detail.

- Explicitly indicate which commands work and which do not and the order in which commands should be run.

- Document any errors encountered as well as the steps taken to workaround them.

- Document any other steps or information that the agent can use to reduce time spent exploring or trying and failing to run bash commands.

- Finally, explicitly instruct the agent to trust the instructions and only perform a search if the information in the instructions is incomplete or found to be in error.

</StepsToFollow>

- Document any errors encountered as well as the steps taken to work-around them.

1. Click **Send now** or press Return.

Copilot will start a new session, which will appear in the list below the prompt box. Copilot will create a draft pull request, write your custom instructions, push them to the branch, then add you as a reviewer when it has finished, triggering a notification.

[**Writing your own .github/copilot-instructions.md file**](https://docs.github.com/en/copilot/how-tos/configure-custom-instructions/add-repository-instructions#writing-your-own-githubcopilot-instructionsmd-file)

1. In the root of your repository, create a file named .github/copilot-instructions.md.

Create the .github directory if it does not already exist.

1. Add natural language instructions to the file, in Markdown format.

Whitespace between instructions is ignored, so the instructions can be written as a single paragraph, each on a new line, or separated by blank lines for legibility.

Tip

The first time you create a pull request in a given repository with Copilot coding agent, Copilot will leave a comment with a link to automatically generate custom instructions for the repository.

[**Using one or more .instructions.md files**](https://docs.github.com/en/copilot/how-tos/configure-custom-instructions/add-repository-instructions#using-one-or-more-instructionsmd-files-1)

1. Create the .github/instructions directory if it does not already exist.
2. Create one or more .instructions.md files, adding natural language instructions to the file(s).

Whitespace between instructions is ignored, so the instructions can be written as a single paragraph, each on a new line, or separated by blank lines for legibility.

1. Specify what files or directories the instructions apply to by adding applyTo frontmatter to the Markdown files, using glob syntax.
2. ---
3. **applyTo: "app/models/\*\*/*\*.rb"***
4. ***---***
5. ***Add custom instructions here***

To apply the instructions to all files, use the \*\* pattern.

Did you successfully add a custom instructions file to your repository?

[**Writing effective repository custom instructions**](https://docs.github.com/en/copilot/how-tos/configure-custom-instructions/add-repository-instructions#writing-effective-repository-custom-instructions)

The instructions you add to your custom instruction file(s) should be short, self-contained statements that provide Copilot with relevant information to help it work in this repository. Because the instructions are sent with every chat message, they should be broadly applicable to most requests you will make in the context of the repository.

The exact structure you utilize for your instructions file(s) will vary by project and need, but the following guidelines provide a good starting point:

* Provide an overview of the project you're working on, including its purpose, goals, and any relevant background information.
* Include the folder structure of the repository, including any important directories or files that are relevant to the project.
* Specify the coding standards and conventions that should be followed, such as naming conventions, formatting rules, and best practices.
* Include any specific tools, libraries, or frameworks that are used in the project, along with any relevant version numbers or configurations.

The following instructions file(s) is an example of these practices in action:

**# Project Overview**

This project is a web application that allows users to manage their tasks and to-do lists. It is built using React and Node.js, and uses MongoDB for data storage.

**## Folder Structure**

- `/src`: Contains the source code for the frontend.

- `/server`: Contains the source code for the Node.js backend.

- `/docs`: Contains documentation for the project, including API specifications and user guides.

**## Libraries and Frameworks**

- React and Tailwind CSS for the frontend.

- Node.js and Express for the backend.

- MongoDB for data storage.

**## Coding Standards**

- Use semicolons at the end of each statement.

- Use single quotes for strings.

- Use function based components in React.

- Use arrow functions for callbacks.

**## UI guidelines**

- A toggle is provided to switch between light and dark mode.

- Application should have a modern and clean design.

You should also consider the size and complexity of your repository. The following types of instructions may work for a small repository with only a few contributors, but for a large and diverse repository, **these may cause problems**:

* Requests to refer to external resources when formulating a response
* Instructions to answer in a particular style
* Requests to always respond with a certain level of detail

For example, the following instructions **may not have the intended results**:

Always conform to the coding styles defined in styleguide.md in repo my-org/my-repo when generating code.

Use @terminal when answering questions about Git.

Answer all questions in the style of a friendly colleague, using informal language.

Answer all questions in less than 1000 characters, and words of no more than 12 characters.

[**Repository custom instructions in use**](https://docs.github.com/en/copilot/how-tos/configure-custom-instructions/add-repository-instructions#repository-custom-instructions-in-use)

The instructions in the file(s) are available for use by Copilot Chat as soon as you save the file(s). The complete set of instructions will be automatically added to requests that you submit to Copilot in the context of that repository. For example, they are added to the prompt you submit to Copilot Chat.

In Copilot Chat's immersive view ([github.com/copilot](https://github.com/copilot)), you can start a conversation that uses repository custom instructions by adding, as an attachment, the repository that contains the instructions file.

Whenever repository custom instructions are used by Copilot Chat, the instructions file is added as a reference for the response that's generated. To find out whether repository custom instructions were used, expand the list of references at the top of a chat response in the Chat panel and check whether the .github/copilot-instructions.md file is listed.



You can click the reference to open the file.

Note

* It is possible for multiple types of custom instructions to apply to a conversation. Personal instructions take the highest priority, followed by repository instructions, with organization instructions prioritized last. However, all sets of relevant instructions are still combined and provided to Copilot Chat.
* Whenever possible, you should avoid providing conflicting sets of instructions. If you are concerned about response quality, you can also choose to temporarily disable repository instructions. See [Adding repository custom instructions for GitHub Copilot](https://docs.github.com/en/copilot/customizing-copilot/adding-repository-custom-instructions-for-github-copilot?tool=webui#enabling-or-disabling-repository-custom-instructions).

[**Enabling or disabling repository custom instructions**](https://docs.github.com/en/copilot/how-tos/configure-custom-instructions/add-repository-instructions#enabling-or-disabling-repository-custom-instructions)

You can choose whether or not you want Copilot to use repository-based custom instructions.

[**Enabling or disabling custom instructions for Copilot Chat**](https://docs.github.com/en/copilot/how-tos/configure-custom-instructions/add-repository-instructions#enabling-or-disabling-custom-instructions-for-copilot-chat)

Custom instructions are enabled for Copilot Chat by default but you can disable, or re-enable, them at any time. This applies to your own use of Copilot Chat and does not affect other users.

1. On GitHub.com, do one of the following:
   * Go to a repository with a custom instructions file and open the assistive chat panel.
   * Go to the immersive view of Copilot Chat ([github.com/copilot](https://github.com/copilot)) and attach a repository that contains a custom instructions file.
2. Click the  button at the top of the Chat panel, or the top right of the immersive page.
3. Click **Disable custom instructions** or **Enable custom instructions**.

Note

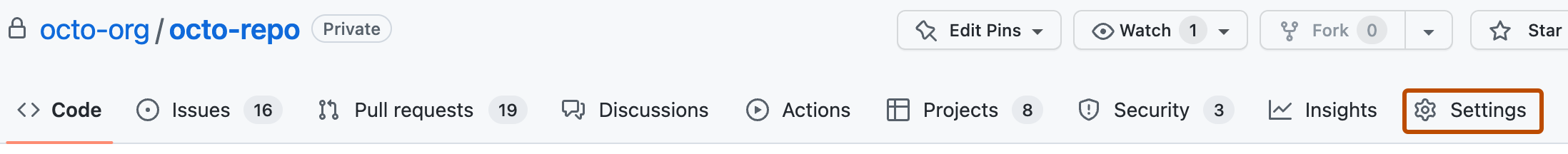
You will only see these options in the context of a repository that contains a custom instructions file.

Your choice persists, for all repositories containing a custom instructions file, until you change it.

[**Enabling or disabling custom instructions for Copilot code review**](https://docs.github.com/en/copilot/how-tos/configure-custom-instructions/add-repository-instructions#enabling-or-disabling-custom-instructions-for-copilot-code-review)

Custom instructions are enabled for Copilot code review by default but you can disable, or re-enable, them in the repository settings on GitHub.com. This applies to Copilot's use of custom instructions for all code reviews it performs in this repository.

1. On GitHub, navigate to the main page of the repository.
2. Under your repository name, click **Settings**. If you cannot see the "Settings" tab, select the  dropdown menu, then click **Settings**.



1. In the "Code & automation" section of the sidebar, click **Copilot**, then **Code review**.
2. Toggle the “Use custom instructions when reviewing pull requests” option on or off.