The board layout spreads your issues, pull requests, and draft issues across customizable columns. You can create a kanban board by setting your column field to a "Status" field or set any other single select or iteration field as the column field. You can drag individual or multiple items from column to column and the value of those items will adjust to match the column you drag them to. For more information about changing a view to use the board layout, see Changing the layout of a view. Setting a limit on the number of items in a column You can set a limit for the number of cards in a particular column in a board layout. Setting a limit does not restrict anyone from adding cards that would exceed the column's limit, nor does it restrict any automations from adding cards. Column limits are unique to each view in your project. The current count of cards and the column's limit is displayed at the top of the column and is highlighted when the current count exceeds the limit. You can use column limits to communicate how you want the column to be used and to make clear at which point action needs to be taken to reduce the number of the cards in the column. Click next to the name of the column you want to modify. In the menu, click Set column limit. Under "Column limit", type the card limit for this column. Optionally, to remove the limit, clear the entry. Click Save. Showing and hiding fields Click next to the name of the currently open view. Under "Configuration", click Fields. Select or deselect the fields you want to show or hide. Setting the column field in board layout In the board layout, you choose any single select or iteration field for your columns. If you drag an item to a new column, the value of that column is applied to the dragged item. For example, if you use the "Status" field for your board columns and then drag an item with a status of In progress to the Done column, the status of the item will switch to Done. Click next to the name of the currently open view. Click Column field. Click the field you want to use. Showing and hiding columns in board layout In the board layout, you can choose which columns to display. The available columns are made up of the contents of your selected column field. In the board layout, scroll to the right of your columns, and click . Select the columns you want to show. Slicing by field values You can slice your items by a field to view a list of the field values in a separate panel. When you click on a value in the slice panel, the current view will adjust to only show items with that value. The slice panel works with the current filter applied to your view. Note You cannot slice by title, reviewers, or linked pull requests. Click next to the name of the currently open view. Click Slice by. Click the field you want to slice by. Optionally, to disable slicing, click No slicing at the bottom of the list. With the slice panel open, you can change the current field by using the menu at the top of the slice panel. Sorting by field values You can sort items by a field value. Note When a board is sorted, you cannot manually reorder items within a column. You can continue to move items from column to column. Click next to the name of the currently open view. Click Sort. Click the field you want to sort by. Optionally, to add a secondary sort, click another field. Optionally, to change the direction of the sort for either field, click the field again. Optionally, to remove a sort, click one of the sorted fields, or click No sorting at the bottom of the list. Grouping by field values You can use a custom field value to group items and create horizontal sections on your board. These sections provide an additional way to organize and visually separate items. Additionally, horizontal grouping allows you to differentiate work according to work streams, team members, or varying levels of urgency or priority. When items are grouped, if you drag an item to a new group, the value of that group is applied. For example, if you group by "Status" and then drag an item with a status of In progress to the Done group, the status of the item will switch to Done. Similarly, when you add a new item to a group, the new item is populated with the value of the group. Note You cannot group by title, labels, reviewers, or linked pull requests. Click next to the name of the currently open view. Click Group by. Click the field you want to group by. Optionally, to disable grouping, click No grouping at the bottom of the list. Showing the sum of a number field You can configure a view to show the sum of one of more number fields, including a count of items in the group or column. For example, if you have a number field tracking the number of hours each item may take to complete, you can display of sum of those hours for each group or column. In a board layout, field sums are displayed at the top of each column. In table and roadmap layouts, when you enable grouping by a field, field sums are included in each group's header. Click next to the name of the currently open view. Click Field sum. Select the fields you want to include. GitHub Copilot supports multiple AI models with different capabilities. The model you choose affects the quality and relevance of responses by Copilot Chat and Copilot code completion. Some models offer lower latency, while others offer fewer hallucinations or better performance on specific tasks. This article helps you compare the available models, understand the strengths of each model, and choose the model that best fits your task. For guidance across different models using real-world tasks, see Comparing AI models using different tasks. The best model depends on your use case: For balance between cost and performance, try GPT-4.1 or Claude Sonnet 3.7. For fast, low-cost support for basic tasks, try o4-mini or Claude Sonnet 3.5. For deep reasoning or complex coding challenges, try o3, GPT-4.5, or Claude Sonnet 3.7. For multimodal inputs and real-time performance, try Gemini 2.0 Flash or GPT-4.1. You can click a model name in the list below to jump to a detailed overview of its strengths and use cases. GPT-4.1 OpenAI's latest model, GPT-4.1, is now available in GitHub Copilot and GitHub Models, bringing OpenAI's newest model to your coding workflow. This model outperforms GPT-4o across the board, with major gains in coding, instruction following, and long-context understanding. It has a larger context window and features a refreshed knowledge cutoff of June 2024. OpenAI has optimized GPT-4.1 for real-world use based on direct developer feedback about: frontend coding, making fewer extraneous edits, following formats reliably, adhering to response structure and ordering, consistent tool usage, and more. This model is a strong default choice for common development tasks that benefit from speed, responsiveness, and general-purpose reasoning. Use cases GPT-4.1 is a revamped version of OpenAI's GPT-4o model. This model is a strong default choice for common development tasks that benefit from speed, responsiveness, and general-purpose reasoning. If you're working on tasks that require broad knowledge, fast iteration, or basic code understanding, GPT-4.1 makes large improvements over GPT-4o. Strengths The following table summarizes the strengths of GPT-4.1: Alternative options TaskDescriptionWhy another model may be betterMulti-step reasoning or algorithmsDesign complex logic or break down multi-step problems.GPT-4.5 or Claude Sonnet 3.7 provide better step-by-step thinking.Complex refactoringRefactor large codebases or update multiple interdependent files.GPT-4.5 handles context and code dependencies more robustly.System review or architectureAnalyze structure, patterns, or architectural decisions in depth.Claude Sonnet 3.7 or GPT-4.5 offer deeper analysis. GPT-4o OpenAI GPT-4o is a multimodal model that supports text and images. It responds in real time and works well for lightweight development tasks and conversational prompts in Copilot Chat. Compared to previous models, GPT-4o improves performance in multilingual contexts and demonstrates stronger capabilities when interpreting visual content. It delivers GPT-4 Turbo–level performance with lower latency and cost, making it a good default choice for many common developer tasks. For more information about GPT-4o, see OpenAI's documentation. Use cases GPT-4o is a good choice for common development tasks that benefit from speed, responsiveness, and general-purpose reasoning. If you're working on tasks that require broad knowledge, fast iteration, or basic code understanding, GPT-4o is likely the model to use. Strengths The following table summarizes the strengths of GPT-4o: Alternative options The following table summarizes when an alternative model may be a better choice: GPT-4.5 Note GPT-4.5 in Copilot Chat is currently in public preview and subject to change. OpenAI GPT-4.5 improves reasoning, reliability, and contextual understanding. It works well for development tasks that involve complex logic, high-quality code generation, or interpreting nuanced intent. Compared to GPT-4.1, GPT-4.5 produces more consistent results for multi-step reasoning, long-form content, and complex problem-solving. It may have slightly higher latency and costs than GPT-4.1 and other smaller models. For more information about GPT-4.5, see OpenAI's documentation. Use cases GPT-4.5 is a good choice for tasks that involve multiple steps, require deeper code comprehension, or benefit from a conversational model that handles nuance well. Strengths The following table summarizes the strengths of GPT-4.5: Alternative options The following table summarizes when an alternative model may be a better choice: o1 OpenAI o1 is an older reasoning model that supports complex, multi-step tasks and deep logical reasoning to find the best solution. For more information about o1, see OpenAI's documentation. Use cases o1 is a good choice for tasks that require deep logical reasoning. Its ability to reason through complex logic enables Copilot to break down problems into clear, actionable steps. This makes o1 particularly well-suited for debugging. Its internal reasoning can extend beyond the original prompt to explore the broader context of a problem and can uncover edge cases or root causes that weren't explicitly mentioned. Strengths The following table summarizes the strengths of o1: Alternative options The following table summarizes when an alternative model may be a better choice: o3 Note o3 in Copilot Chat is currently in public preview and subject to change. OpenAI o3 is the most capable reasoning model in the o-series. It is ideal for deep coding workflows and complex, multi-step tasks. For more information about o3, see OpenAI's documentation. Use cases o3 is a good choice for tasks that require deep logical reasoning. Its ability to reason through complex logic enables Copilot to break down problems into clear, actionable steps. This makes o3 particularly well-suited for debugging. Its internal reasoning can extend beyond the original prompt to explore the broader context of a problem and can uncover edge cases or root causes that weren't explicitly mentioned. Strengths The following table summarizes the strengths of o3: Alternative options The following table summarizes when an alternative model may be a better choice: o3-mini OpenAI o3-mini is a fast, cost-effective reasoning model designed to deliver coding performance while maintaining lower latency and resource usage. o3-mini outperforms o1 on coding benchmarks with response times that are comparable to o1-mini. Copilot is configured to use OpenAI's "medium" reasoning effort. For more information about o3-mini, see OpenAI's documentation. Use cases o3-mini is a good choice for developers who need fast, reliable answers to simple or repetitive coding questions. Its speed and efficiency make it ideal for lightweight development tasks. Strengths The following table summarizes the strengths of o3-mini: Alternative options The following table summarizes when an alternative model may be a better choice: o4-mini Note o4-mini in Copilot Chat is currently in public preview and subject to change. OpenAI o4-mini is the most efficient model in the o-series. It is a cost-effective reasoning model designed to deliver coding performance while maintaining lower latency and resource usage. For more information about o4, see OpenAI's documentation. Use cases o4-mini is a good choice for developers who need fast, reliable answers to simple or repetitive coding questions. Its speed and efficiency make it ideal for lightweight development tasks. Strengths The following table summarizes the strengths of o4-mini: Alternative options The following table summarizes when an alternative model may be a better choice: Claude Sonnet 3.5 Claude Sonnet 3.5 is a fast and cost-efficient model designed for everyday developer tasks. While it doesn't have the deeper reasoning capabilities of Claude Sonnet 3.7, it still performs well on coding tasks that require quick responses, clear summaries, and basic logic. For more information about Claude Sonnet 3.5, see Anthropic's documentation. For more information on using Claude in Copilot, see Using Claude in Copilot Chat. Use cases Claude Sonnet 3.5 is a good choice for everyday coding support—including writing documentation, answering language-specific questions, or generating boilerplate code. It offers helpful, direct answers without over-complicating the task. If you're working within cost constraints, Claude Sonnet 3.5 is recommended as it delivers solid performance on many of the same tasks as Claude Sonnet 3.7, but with significantly lower resource usage. Strengths The following table summarizes the strengths of Claude Sonnet 3.5: Alternative options The following table summarizes when an alternative model may be a better choice: Claude Sonnet 3.7 Claude Sonnet 3.7 is a powerful model that excels in development tasks that require structured reasoning across large or complex codebases. Its hybrid approach to reasoning responds quickly when needed, while still supporting slower step-by-step analysis for deeper tasks. For more information about Claude Sonnet 3.7, see Anthropic's documentation. For more information on using Claude in Copilot, see Using Claude in Copilot Chat. Use cases Claude Sonnet 3.7 excels across the software development lifecycle, from initial design to bug fixes, maintenance to optimizations. It is particularly well-suited for multi-file refactoring or architectural planning, where understanding context across components is important. Strengths The following table summarizes the strengths of Claude Sonnet 3.7: Alternative options The following table summarizes when an alternative model may be a better choice: Claude Sonnet 4 Note Claude Sonnet 4 in Copilot Chat is currently in public preview and subject to change. For more information about Claude Sonnet 4, see Anthropic's documentation. For more information on using Claude in Copilot, see Using Claude in Copilot Chat. Claude Opus 4 Note Claude Opus 4 in Copilot Chat is currently in public preview and subject to change. For more information about Claude Opus 4, see Anthropic's documentation. For more information on using Claude in Copilot, see Using Claude in Copilot Chat. Gemini 2.0 Flash Gemini 2.0 Flash is Google's high-speed, multimodal model optimized for real-time, interactive applications that benefit from visual input and agentic reasoning. In Copilot Chat, Gemini 2.0 Flash enables fast responses and cross-modal understanding. For more information about Gemini 2.0 Flash, see Google's documentation. For more information on using Gemini in Copilot, see Using Gemini in Copilot Chat. Use cases Gemini 2.0 Flash supports image input so that developers can bring visual context into tasks like UI inspection, diagram analysis, or layout debugging. This makes Gemini 2.0 Flash particularly useful for scenarios where image-based input enhances problem-solving, such as asking Copilot to analyze a UI screenshot for accessibility issues or to help understand a visual bug in a layout. Strengths The following table summarizes the strengths of Gemini 2.0 Flash: Alternative options The following table summarizes when an alternative model may be a better choice: Gemini 2.5 Pro Note Gemini 2.5 Pro in Copilot Chat is currently in public preview and subject to change. Gemini 2.5 Pro is Google's latest AI model, designed to handle complex tasks with advanced reasoning and coding capabilities. It also works well for heavy research workflows that require long-context understanding and analysis. For more information about Gemini 2.5 Pro, see Google's documentation. For more information on using Gemini in Copilot, see Using Gemini in Copilot Chat. Use cases Gemini 2.5 Pro is well-suited for advanced coding tasks, such as developing complex algorithms or debugging intricate codebases. It can assist with scientific research by analyzing data and generating insights across a wide range of disciplines. Its long-context capabilities allow it to manage and understand extensive documents or datasets effectively. Gemini 2.5 Pro is a strong choice for developers needing a powerful model. Strengths The following table summarizes the strengths of Gemini 2.5 Pro: Alternative options The following table summarizes when an alternative model may be a better choice: Further reading You can't perform that action at this time. Page 2 You can't perform that action at this time. You can't perform that action at this time. Page 2 You can't perform that action at this time. You can't perform that action at this time. A project is an adaptable spreadsheet, task-board, and road map that integrates with your issues and pull requests on GitHub to help you plan and track your work effectively. You can create and customize multiple views by filtering, sorting, grouping your issues and pull requests, visualize work with configurable charts, and add custom fields to track metadata specific to your team. Rather than enforcing a specific methodology, a project provides flexible features you can customize to your team's needs and processes. To get started and create a project, see Creating a project. To learn more about the different layouts, see Changing the layout of a view. Your projects are built from the issues and pull requests you add, creating direct references between your project and your work. Information is synced automatically to your project as you make changes, updating your views and charts. This integration works both ways, so that when you change information about a pull request or issue in your project, the pull request or issue reflects that information. For example, change an assignee in your project and that change is shown in your issue. You can take this integration even further, group your project by assignee, and make changes to issue assignment by dragging issues into the different groups. To learn more about managing items in your project, see Adding items to your project and Editing items in your project. You can use custom fields to add metadata to your issues, pull requests, and draft issues and build a richer view of item attributes. You're not limited to the built-in metadata (assignee, milestone, labels, etc.) that currently exists for issues and pull requests. For example, you can add the following metadata as custom fields: A date field to track target ship dates. A number field to track the complexity of a task. A single select field to track whether a task is Low, Medium, or High priority. A text field to add a quick note. An iteration field to plan work week-by-week, including support for breaks. To learn more about the different fields you can add to a project, see Understanding fields. Automating your projects There are a number of ways you can add automation to your project. Built-in workflows allow you to automatically set fields when items are added or changed, and you can also configure your project to automatically archive items when they meet certain criteria and automatically add items from a repository when they match set criteria. For more information, see Using the built-in automations. You can also use the GraphQL API and GitHub Actions to take even greater control of your project. For more information, see Using the API to manage Projects and Automating Projects using Actions. Viewing your project from different perspectives Quickly answer your most pressing questions by tailoring your project view to give you the information you need. You can save these views, allowing you to quickly return to them as needed and make them available to your team. Views not only let you scope down the items listed but also offer three different layout options. You can view your project as a high-density table layout, as a kanban board, or a timeline-style roadmap. For more information about the different layout options, see Changing the layout of a view. This project brings support for the ESP8266 chip to the Arduino environment. It lets you write sketches, using familiar Arduino functions and libraries, and run them directly on ESP8266, with no external microcontroller required. ESP8266 Arduino core comes with libraries to communicate over WiFi using TCP and UDP, set up HTTP, mDNS, SSDP, and DNS servers, do OTA updates, use a file system in flash memory, and work with SD cards, servos, SPI and I2C peripherals. Starting with 1.6.4, Arduino allows installation of third-party platform packages using Boards Manager. We have packages available for Windows, Mac OS, and Linux (32 and 64 bit). Download and install Arduino IDE 1.x or 2.x Start Arduino and open the Preferences window Enter into the File>Preferences>Additional Boards Manager URLs field of the Arduino IDE. You can add multiple URLs, separating them with commas. Open Boards Manager from Tools > Board menu and install esp8266 platform (and don't forget to select your ESP8266 board from Tools > Board menu after installation). Boards manager link: Documentation: Also known as latest git or master branch. PlatformIO is an open source ecosystem for IoT development with a cross-platform build system, a library manager, and full support for Espressif (ESP8266) development. It works on the following popular host operating systems: macOS, Windows, Linux 32/64, and Linux ARM (like Raspberry Pi, BeagleBone, CubieBoard). What is PlatformIO? PlatformIO IDE PlatformIO Core (command line tool) Advanced usage - custom settings, uploading to SPIFFS, Over-the-Air (OTA), staging version Integration with Cloud and Standalone IDEs - Cloud9, Codeanywhere, Eclipse Che (Codenvy), Atom, CLion, Eclipse, Emacs, NetBeans, Qt Creator, Sublime Text, VIM, Visual Studio, and VSCode Project Examples makeEspArduino is a generic makefile for any ESP8266 Arduino project. Using make instead of the Arduino IDE makes it easier to do automated and production builds. Documentation for latest development version: ESP8266 Community Forum is a well-established community for questions and answers about Arduino for ESP8266. Stackoverflow is also an alternative. If you need help, have a "How do I..." type question, have a problem with a 3rd party library not hosted in this repo, or just want to discuss how to approach a problem, please ask there. If you find the forum useful, please consider supporting it with a donation. If you encounter an issue which you think is a bug in the ESP8266 Arduino Core or the associated libraries, or if you want to propose an enhancement, you are welcome to submit it here on Github: . Please provide as much context as possible, as well as the information requested in the issue template: ESP8266 Arduino core version which you are using (you can check it in Boards Manager) your sketch code; please wrap it into a code block, see Github markdown manual when encountering an issue that happens at run time, attach the serial output. Wrap it into a code block, just like the code. for issues that happen at compile time, enable verbose compiler output in the IDE preferences, and attach that output (also inside a code block) ESP8266 development board model IDE settings (board choice, flash size) etc For minor fixes of code and documentation, please go ahead and submit a pull request. A gentle introduction to the process can be found here. Check out the list of issues that are easy to fix — easy issues pending. Working on them is a great way to move the project forward. Larger changes (rewriting parts of existing code from scratch, adding new functions to the core, adding new libraries) should generally be discussed by opening an issue first. PRs with such changes require testing and approval. Feature branches with lots of small commits (especially titled "oops", "fix typo", "forgot to add file", etc.) should be squashed before opening a pull request. At the same time, please refrain from putting multiple unrelated changes into a single pull request. Arduino IDE is developed and maintained by the Arduino team. The IDE is licensed under GPL. ESP8266 core includes an xtensa gcc toolchain, which is also under GPL. Esptool.py was initially created by Fredrik Ahlberg (@themadinventor, @kongo), and is currently maintained by Angus Gratton (@projectgus) under GPL 2.0 license. Espressif's NONOS SDK included in this build is under Espressif MIT License. ESP8266 core files are licensed under LGPL. SPI Flash File System (SPIFFS) written by Peter Andersson is used in this project. It is distributed under the MIT license. umm_malloc memory management library written by Ralph Hempel is used in this project. It is distributed under the MIT license. SoftwareSerial library and examples written by Peter Lerup. Distributed under LGPL 2.1. BearSSL library written by Thomas Pornin, built from is used in this project. It is distributed under the MIT License. LittleFS library written by ARM Limited and released under the BSD 3-clause license. uzlib library written and (c) 2014-2018 Paul Sokolovsky, licensed under the ZLib license (. uzlib is based on: tinf library by Joergen Ibsen (Deflate decompression); Deflate Static Huffman tree routines by Simon Tatham; LZ77 compressor by Paul Sokolovsky; with library integrated and maintained by Paul Sokolovsky. Toolchain repo Lwip link layer repo SoftwareSerial repo Serial Monitor Arduino IDE plugin Original discussion here, quick download there. FTP Client/Server Library GitHub IssuesCreate issues, break them into sub-issues, track progress, add custom fields, and have conversations. Visualize large projects as tables, boards, or roadmaps, and automate everything with code.Start using projectsContact salesTackle complex issues with sub-issues and track their status with progress indicators. Navigate the full scope of work all in one view.Express ideas with GitHub Flavored Markdown, mention contributors, react with emoji, clarify with attachments, and see references from commits, pull requests, releases, and deploys. Coordinate by assigning contributors and teams, or by adding them to milestones and projects. All in a single timeline.Bored of boards? Switch to tables and roadmaps. Create views for how you work. Track metadata like iterations, priority, story points, dates, notes, and links. Add custom fields to projects and edit from the issue sidebar. Track the health of your current iteration cycle, milestone, or any other custom field you create with new project insights. Identify bottlenecks and issues blocking the team from making progress with the new burn up chart.Create templates to share and reuse when getting started with a new project. Share inspiration across teams and get started with a single click.Accelerate your project planning with workflows. Automatically triage issues, set values for custom fields, or archive issues. Issues can be viewed, created, and managed in your browser, your favorite terminal, or on your phone or tablet.View, update, and create issues without ever leaving your terminal.Create and manage issues on the go with our native iOS and Android mobile apps.The new planning and tracking functionality keeps my project management close to my code. I no longer find myself needing to reach for spreadsheets or 3P tools which go stale instantly.We all need a way to plan our work, track issues, and discuss the things we build. Our answer to this universal question is GitHub Issues, and it's built-in to every repository. GitHub's issue tracking is unique because of our focus on simplicity, references, and elegant formatting.With GitHub Issues, you can express ideas with GitHub Flavored Markdown, assign and mention contributors, react with emojis, clarify with attachments and videos, plus reference code like commits, pull requests, and deploys. With task lists, you can break big issues into tasks, further organize your work with milestones and labels, and track relationships and dependencies.We built GitHub Issues for developers. It is simple, adaptable, and powerful.As teams and projects grow, how we work evolves. Tools that hard-code a methodology are too specific and rigid to adapt to any moment. Often, we find ourselves creating a spreadsheet or pulling out a notepad to have the space to think. Then our planning is disconnected from where the work happens.The new Projects connect your planning directly to the work your teams are doing and flexibly adapt to whatever your team needs at any point. Built like a spreadsheet, project tables give you a live canvas to filter, sort, and group issues and pull requests. You can use it, or the accompanying project board, along with custom fields, to track a sprint, plan a feature, or manage a large-scale release.All users have access to the free tier of GitHub Issues and Projects. For more information about paid tiers, see our pricing page.Yes! GitHub Enterprise Server (GHES) support follows our regular cadence of one to two quarters before enabling the on-premises functionality. You can't perform that action at this time.