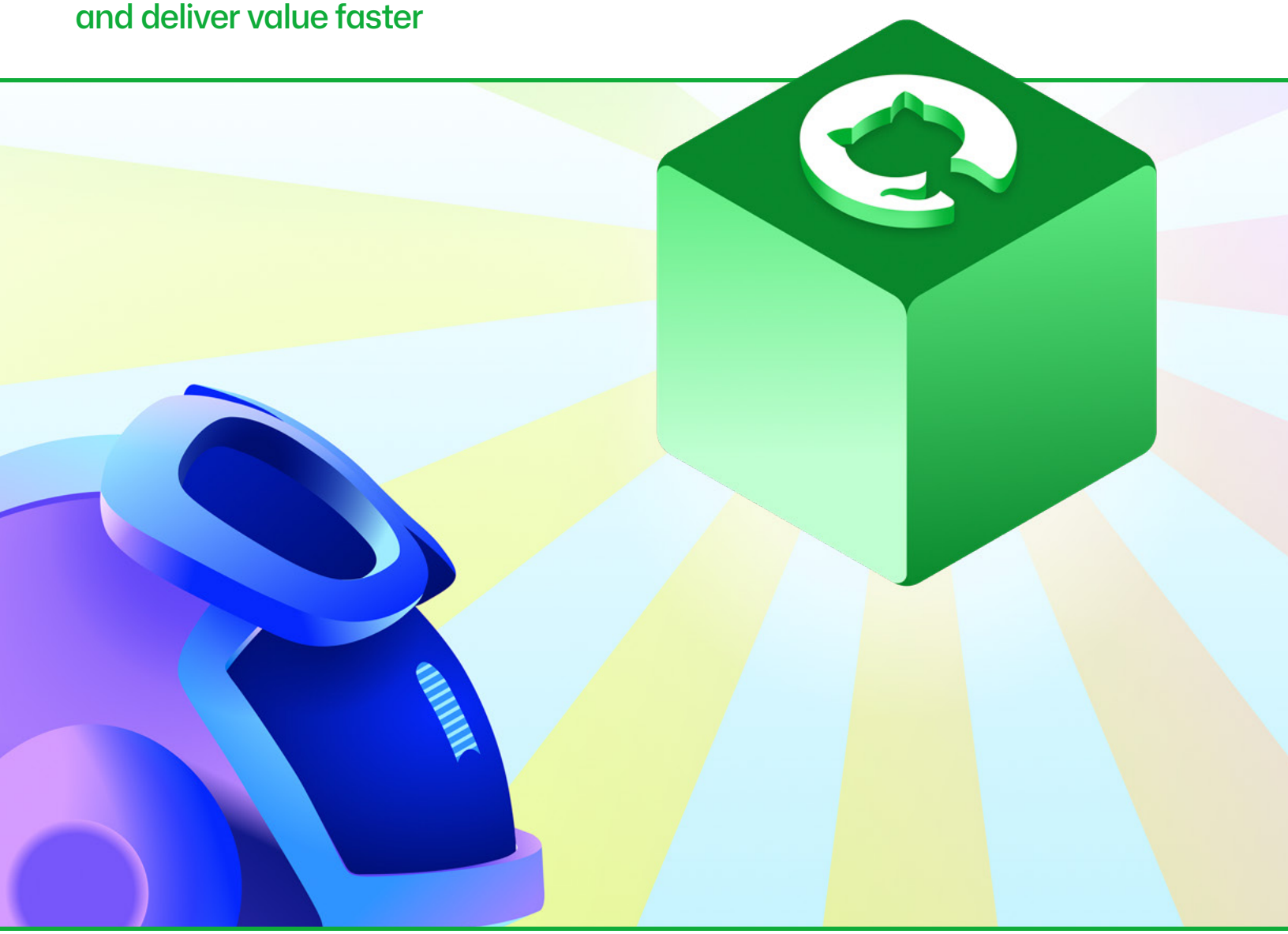
AI-powered DevOps with GitHub and Azure

Boost efficiency, enhance security, and deliver value faster



WRITTEN BY GITHUB WITH

What's inside

What is DevOps?

When implemented effectively, DevOps can transform how your organization delivers software—accelerating release cycles, improving reliability, and driving innovation. The real opportunity lies in how DevOps enables you to stay agile in a rapidly evolving market. By establishing a culture of collaboration, continuous improvement, and strategic technology adoption, you can outpace the competition with faster time to market and a stronger ability to adapt to change.

DevOps is shaped by diverse experiences, technical skills, and cultural perspectives. This diversity brings about multiple interpretations and evolving practices, making DevOps a dynamic and interdisciplinary field. A DevOps team is cross-functional and involves key players from teams that are part of the software delivery lifecycle (SDLC).

In this ebook, we will explore the value of building a strong DevOps team and practice, and how to apply AI to automate routine tasks, protect code, and achieve optimal end-to-end lifecycle management.

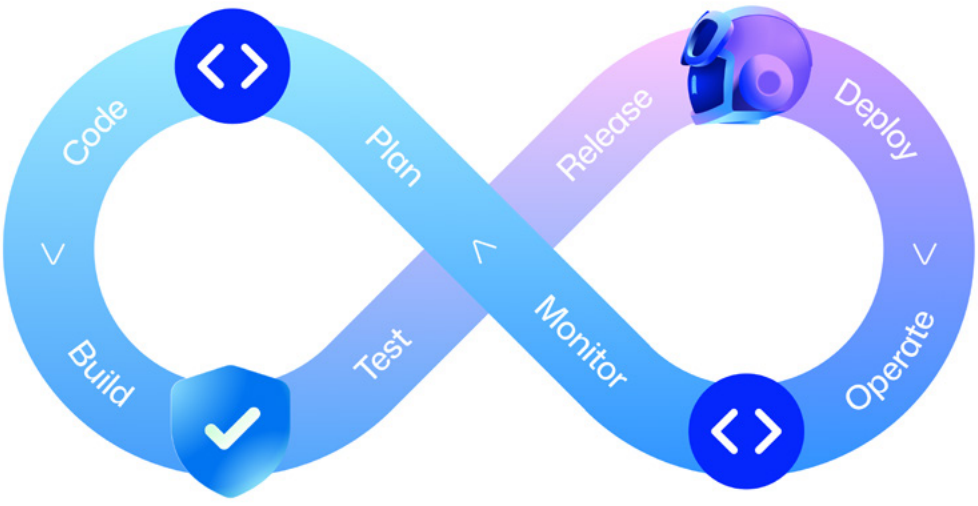


Figure 1: Continuous DevOps lifecycle

DevOps defined

Donovan Brown, a trusted voice in the DevOps community, shared a definition of DevOps that has been widely recognized by DevOps practitioners:

> "DevOps is the union of people, process, and products to enable continuous delivery of value to your end users."

Donovan Brown // Partner Program Manager, Microsoft[1]

In many tech environments, teams are siloed by their technical skill sets, with each focusing on their own metrics, KPIs, and deliverables. This fragmentation often slows down delivery, causes inefficiencies, and leads to conflicting priorities, ultimately hindering progress. Today 75% of developers lose between six and 15 hours weekly due to tool sprawl according to the 2025 State of Internal Developer Portals report. In a crowded market, AI solutions that deliver tangible results will win.

To overcome these challenges, organizations should work to foster collaboration, encourage constructive feedback, automate workflows, and embrace continuous improvement. This helps ensure faster software delivery, greater efficiency, improved decision making, cost savings, and a stronger competitive edge.

How can teams begin adopting new DevOps practices effectively? They can start by addressing the most significant pain points first, such as manual deployment processes, long feedback cycles, inefficient test automation, and delays caused by manual interventions in release pipelines.

Eliminating friction points can feel overwhelming, but the rapid rise of AI in recent years has created new opportunities for developers to increase the speed and quality of their work. Our research found that the quality of the code authored and

1: https://www.donovanbrown.com/post/what-is-devops

reviewed was better across the board when using GitHub Copilot Chat, even when none of the developers had used the AI-powered tool before.

- Developers are already writing code over **55% faster** with Copilot.
- **85%** of developers felt more confident in their code quality when authoring code with GitHub Copilot and GitHub Copilot Chat
- Teams leveraging Copilot Autofix reduce Mean Time to Remediation (MTTR) by up to **60%**, helping both security teams and developers stay ahead of risk
- Code reviews were more actionable and completed
- **15% faster** than without GitHub Copilot Chat
- Between **60–75%** of users reported they feel more fulfilled with their job, feel less frustrated when coding, and are able to focus on more satisfying work when using GitHub Copilot.

If AI adoption increases by 25%...

Figure 2: Impacts of AI adoption on developers[2]

2: https://github.blog/news-insights/research/research-quantifying-github-copilots-impact-on-code-quality/

DevOps + agenticAI: Using AI for efficiency

By promoting a culture of shared responsibility, DevOps encourages collaboration and breaks down silos. Agentic AI takes this even further by introducing agents that can automate more complex, multi-step workflows and enable faster feedback cycles—letting teams stay focused on high-value work.

High-performing teams can identify repetitive tasks that hinder productivity and automate them with agents. This enables teams to focus on high-value work while AI acts as a true collaborator on development teams. The ultimate goal is to deliver what matters most to customers and end users while driving organizational growth, accelerating time to market, and bolstering developer productivity and satisfaction.

Automating the mundane

Developers often handle daily tasks that are repetitive. These are commonly referred to as “time thieves” and include things such as manual system checks, setting up new code environments or identifying and addressing bugs. These tasks take time away from a developer’s core responsibility: delivering new features.

DevOps is equal parts team alignment and automation. The overarching goal is to remove burdens and roadblocks from the SDLC and to help developers reduce manual and mundane tasks. Let’s look at how you can utilize agentic AI to resolve these issues.

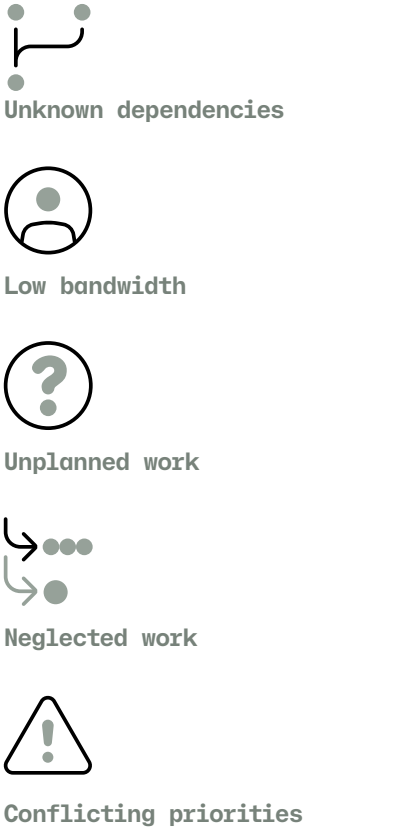


Figure 3: Activities that diminish developer efficiency

Streamline development lifecycles with GitHub

Let's combine DevOps, agentic AI, and the power of GitHub to see how your teams can deliver end-to-end value. GitHub is widely recognized as the home of open-source software, but it also offers enterprise-level features through its GitHub Enterprise solution.

GitHub Enterprise streamlines the DevOps lifecycle by providing a flexible, developer-first platform for version control, issue tracking, and seamless collaboration between teammates and with AI agents. By bringing human and AI collaboration into the same workflow, teams can reduce friction, increase velocity, and stay focused on high-value work without sacrificing visibility or control. With tools like GitHub Copilot and GitHub Actions built in, development teams can automate more, collaborate better, and move faster from idea to production.

With GitHub Copilot, development cycles can be accelerated through multi-file code generation, intelligent refactoring, and automated test creation. With Copilot code review, teams can receive AI-powered feedback on pull requests and ensure consistency and reliability before merging changes. For more complex tasks like feature implementation and multi-step edits across the codebase, GitHub Copilot coding agent can further speed delivery and shorten time to market.

Built-in automation and CI/CD workflows on GitHub, such as GitHub Actions, also help simplify code reviews, testing, and deployment. This reduces the number of manual tasks, while shortening approval times and accelerating development. These tools enable seamless collaboration, breaking down silos and allowing teams to manage every aspect of their projects efficiently—from planning to delivery.

Work smarter, not harder

Automation is at the heart of DevOps, making it possible to eliminate the time thieves and focus on delivering value faster. Automation is a very broad term that includes various items from the SDLC. Automation can include things such as configuring CI/CD to allow for the seamless integration of code changes into your production environment. This can also include automating your infrastructure as code (IaC), testing, monitoring and alerting, and security.

While most DevOps tools provide CI/CD capabilities, GitHub goes a step further with GitHub Actions, a solution that delivers enterprise-grade software to your environment–whether in the cloud, on-premises, or elsewhere. With GitHub Actions, you can not only host your CI/CD pipelines but also automate virtually anything within your workflows. GitHub also supports a broad ecosystem of integrations, so you can connect with the CI/CD tools your team already knows and prefers.

This seamless integration with the GitHub platform eliminates the need for extra tools, streamlining workflows and boosting productivity. Here's how GitHub Actions can transform your workflows:

- **Faster CI/CD:** Automate build, test, and deployment pipelines for quicker releases.
- **Improved code quality:** Enforce code formatting and use Copilot code review to catch bugs, vulnerabilities, and performance issues early.
- **Enhanced collaboration:** Automate notifications and communication around development processes.
- **Simplified compliance**: Helps align repositories with organizational standards.
- **Increased efficiency:** Automate repetitive tasks to free up developers' time.

GitHub Copilot can be used to create GitHub Actions and build intelligent workflows tailored to your needs. It can also suggest coding best practices aligned with your organization's standards, helping teams implement governance and conventions more effectively. GitHub Copilot works across various programming languages and can help author entire workflows or automate tasks at scale.

To learn more about GitHub Copilot, see:

- Getting code suggestions in your IDE with GitHub Copilot
- Using GitHub Copilot in your IDE: tips, tricks, and best practices
- 10 unexpected ways to use GitHub Copilot
- Agent mode 101: All about GitHub Copilot's powerful mode

Reduce repetitive tasks

Focus on automating routine processes and using tools such as GitHub Copilot to streamline your workflow. For example, Copilot can assist with generating unit tests—a time-consuming but essential part of software development. By crafting precise prompts, developers can guide Copilot to create comprehensive testing suites, covering both basic scenarios and more complex edge cases. Teams can also assign pull requests to Copilot for automated review, which can catch bugs and other issues, and even suggest fixes for them, helping teams maintain high code quality."

It is essential to trust, but verify, the results that Copilot provides—much like with any generative AI-powered tool. Your teams can rely on Copilot for simple and complex tasks, but it's important to always validate its output through thorough testing before deploying any code. This not only helps ensure reliability but also prevents errors that could otherwise slow down your workflow.

As you continue using Copilot, refining your prompts will help you make the most of its capabilities, enabling smarter automation while further minimizing repetitive tasks.

For more information on creating unit tests with GitHub Copilot, see:

- Develop unit tests using GitHub Copilot tools
- Writing tests with GitHub Copilot

Prompt engineering and context

Integrating GitHub Copilot into your DevOps practice can revolutionize the way your team works. Crafting precise, context-rich prompts for Copilot can help your team unlock new levels of efficiency and streamline processes.

These benefits can translate into measurable outcomes for your organization, such as:

- **Increased efficiency:** Automate repetitive tasks, minimize manual intervention, and enable faster, smarter decision-making with actionable insights.
- **Cost savings:** Streamline workflows, reduce errors, and lower development costs by integrating AI into repetitive and error-prone processes.
- **Drive results:** Utilize Copilot to support strategic goals, improve customer experiences, and maintain a competitive edge in the market.

By learning how to write precise and detailed prompts, teams can significantly improve the relevance and accuracy of Copilot's suggestions. Like any new tool, proper onboarding and training are essential to help your team maximize Copilot's benefits at scale.

Here's how you can foster a culture of effective prompt engineering within your team:

- **Build an internal community:** Set up chat channels for sharing insights, attend or host events, and create learning opportunities to create a space for your teams to learn.
- **Share surprising moments:** Use tools such as Copilot to create documentation that guides others on their journey.
- **Share tips and tricks that you have picked up:** Host knowledge sharing sessions and use your internal communications (newsletters, Teams, Slack, etc.) to share insights.

Effective prompts help align AI with your team's objectives, which can lead to better decision making, more reliable outputs, and higher performance. By implementing these prompt engineering methods, you can not only save costs but enable faster delivery, enhanced product offerings, and superior customer experiences.

DevOps + security: Protecting code from the inside out

A unified strategy for managing your SDLC is far more effective when it's supported by a streamlined toolset. While tool sprawl is a common challenge across many DevOps disciplines, application security often feels its impact most. Teams frequently add new tools to address gaps, but this approach often overlooks the core issues related to people and processes. As a result, security landscapes can become cluttered with everything from single-application scanners to complex enterprise risk platforms.

By simplifying your toolset, you help developers stay focused, reduce context switching, and maintain their coding flow. A platform where security is integrated at every step—ranging from dependency management and vulnerability alerts to preventive measures that protect sensitive information—brings stability to your organization's software security posture. Additionally, extensibility is crucial, enabling you to utilize your existing tools alongside the platform's built-in capabilities.

Protect every line of code

When you think about software development, languages such as Python, C#, Java, and Rust likely come to mind. However, code takes many forms, and professionals across various fields—data scientists, security analysts, and business intelligence analysts—also engage with coding in their own ways. By extension, your potential risk for security vulnerabilities increases—sometimes unknowingly. Providing a comprehensive set of standards and methodologies to all developers, regardless of their role or title, enables them to integrate security into every step of the cycle.

Static analysis and secret scanning

Using application security testing (AST) tools has become more common when it comes to build-time integration. One minimally invasive technique is to scan the source code as-is, looking for points of complexity, potential exploits, and adherence to standards. The use of software composition analysis (SCA) on

every commit and every push helps developers focus on the task at hand while providing a mechanism for pull requests and code reviews to be more productive and meaningful. With Copilot Autofix, developers can receive automated code suggestions and fixes for vulnerabilities, streamlining remediation and reducing manual effort.

Secret scanning with push protection detects and prevents secret leaks in real-time and blocks sensitive credentials from being pushed to a repository. With a remarkably low false positive rate and approximately 150 service provider integrations, it enables rapid credential revocation and rotation, enhancing developer productivity.

Find and fix vulnerabilities with CodeQL and Copilot Autofix

CodeQL and Copilot Autofix, included with GitHub Code Security, analyze code to identify vulnerabilities, bugs, and other quality issues, and suggest fixes to speed remediation. First, CodeQL builds a database from your codebase through compilation or interpretation and then employs a query language to search for vulnerable patterns. CodeQL also lets you create custom variant databases tailored to specific cases or proprietary use cases relevant to your business. This flexibility enables the development of reusable vulnerability databases that can be used during scans for other applications within your enterprise.

Once CodeQL finds a vulnerability, Copilot Autofix provides a detailed explanation in natural language, together with a suggested fix. Together, CodeQL and Copilot Autofix help developers fix vulnerabilities as they code, reducing the number of escalations to security teams and helping to minimize the occurrences of vulnerabilities leaking to production.

28 minutes

From vulnerability detection to successful remediation[3]

2.4x more precise

Finds leaked secrets with fewer false positives[4]

90% coverage

Copilot Autofix provides code suggestions for nearly 90% of alert types in all supported languages

3: Overall, the median time for developers to use Copilot Autofix to automatically commit the fix for a PR-time alert was 28 minutes, compared to 1.5 hours to resolve the same alerts manually (3x faster). For SQL injection vulnerabilities: 18 minutes compared to 3.7 hours (12x faster). Based on new code scanning alerts found by CodeQL in pull requests (PRs). These are examples; your results will vary.

4: A Comparative Study of Software Secrets Reporting by Secret Detection Tools, Setu Kumar Basak et al., North Carolina State University, 2023

Demystifying the dependency graph

Modern applications can have dozens of directly referenced packages, which can in turn have dozens of more packages as dependencies. This challenge is amplified as enterprises are faced with managing hundreds of repositories with varying levels of dependencies. This makes security a daunting task, as understanding which dependencies are in use across the organization becomes difficult. Adopting a dependency management strategy that tracks repository dependencies, vulnerabilities, and OSS license types reduces risks and helps detect issues before they reach production.

GitHub Enterprise gives users and admins immediate insights into dependency graphs, along with use alerts from Dependabot that flag out-of-date libraries posing potential security risks.

The repository dependency graph consists of:

- **Dependencies:** A complete list of dependencies identified in the repository
- **Dependents:** Any projects or repositories that have a dependency on the repository
- **Dependabot:** Any findings from Dependabot regarding updated versions of your dependencies

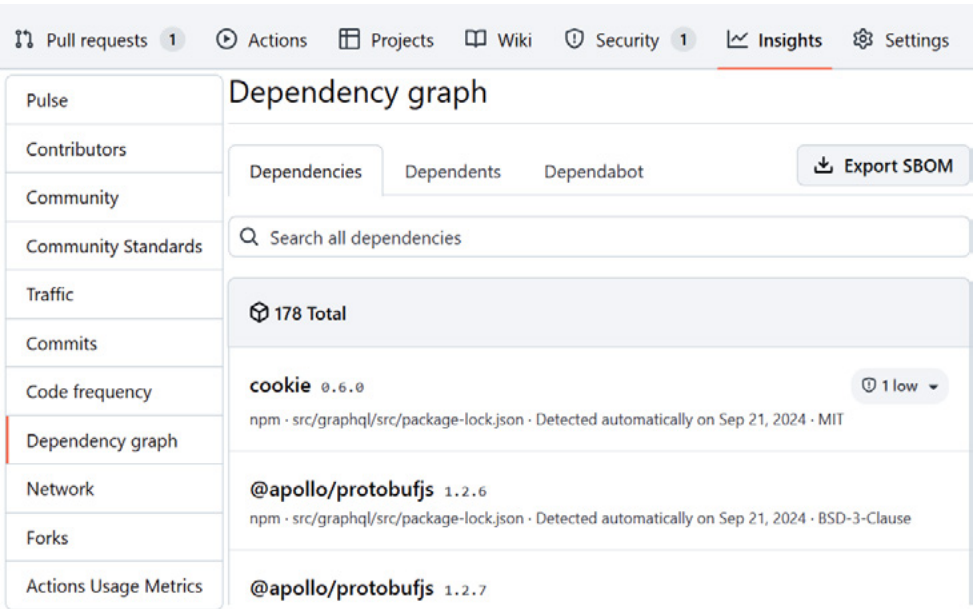


Figure 4: Dependency graph

For repository-level vulnerabilities, the Security tab in the navigation bar shows results for identified vulnerabilities that may be associated with dependencies related to your codebase. The Dependabot view lists alerts related to identified vulnerabilities and allows you to view any rulesets that may help automatically triage certain alerts for public repositories.

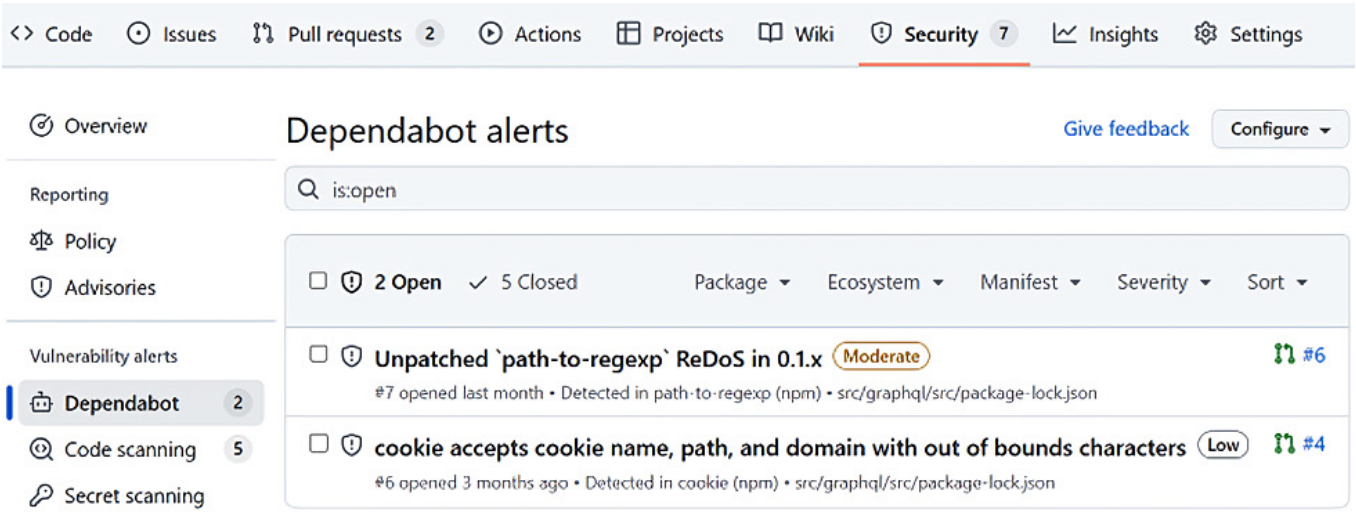


Figure 5: Vulnerability alerts

GitHub Enterprise and organizational views

With GitHub Enterprise, you can view and manage dependencies, vulnerabilities, and OSS licenses across all repositories in your organization and enterprise. The dependency graph allows you to see a comprehensive view of dependencies across all registered repositories.

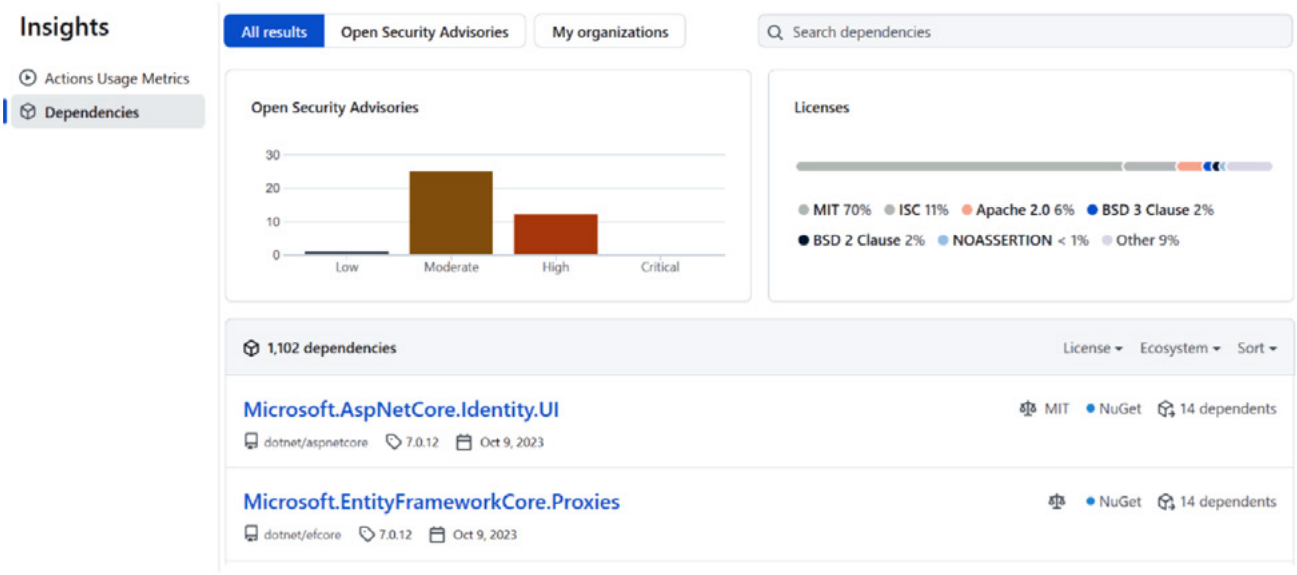


Figure 6: Viewing and managing dependencies, vulnerabilities, and OSS licenses

This at-a-glance dashboard provides an excellent snapshot not only of identified security advisories but also of the distribution of licenses related to dependencies in use across your enterprise. OSS license usage can be particularly risky, especially if you manage proprietary code. Some more restrictive open source licenses, such as GPL and LGPL, can potentially leave your source code vulnerable to forced publication. Open source components require finding a unified way to determine where you may be out of compliance and may wish to find other alternatives for the packages being pulled in with those licenses.

Safeguarding your security posture

Many enterprise-grade source control management systems give you options to safeguard your code using policies, pre-commit hooks, and platform-specific functionality. The following measures can be used to plan out a well-rounded security stance:

Preventive measures: GitHub allows for the configuration and use of different types of rulesets to enforce behaviors and protect against unwanted changes in specific branches. For example:

- Rules requiring pull requests prior to merging changes
- Rules protecting specific branches from having changes pushed directly

An additional client-side check can be performed by using pre-commit hooks. Git, as a source control management system, supports pre-commit hooks to perform various tasks, such as formatting commit messages or running formatting and validation routines before committing changes. These hooks can utilize advanced utilities to help ensure code consistency and quality at the local level.

Protective measures: GitHub allows for configuring protective measures as well, including the use of checks that can be established during a pull request or CI build. These include:

- Dependency checks
- Testing checks
- Code quality checks
- Quality gates
- Manual intervention/human approval gates
- Copilot Autofix

GitHub Enterprise enables software development teams to identify and act on vulnerabilities very quickly, from outdated dependencies and checked-in secrets to known language exploits. With the additional capabilities of viewing the dependency graph, team leaders and admins are armed with the tools they need to stay ahead of the curve when it comes to security advisories. Loop in visibility of the license types in use and you are left with a comprehensive security-first risk management platform.

GitHub Enterprise + Azure: Powering the DevOps pipeline

By now, it's fair to say that the concept of DevOps is widely familiar to those in the technology industry. However, as new tools and methodologies for deploying applications continue to emerge, it can put strain on an ever-growing organization to effectively manage and measure their results.

Meeting the market demands for applications that are resilient, scalable, and cost-effective can be challenging. Utilizing cloud-based resources can help improve time to market, speed up the inner loop for developers, and allow for scaled testing and deployment to occur with cost-conscious controls.

Enabling cloud-native applications

Much like the paradigm of shifting left has brought security, testing, and feedback closer to the development inner loop, the same can be said for developing applications for the cloud. Adopting cloud-centric development practices helps developers bridge the gap between traditional approaches and modern cloud solutions. This shift enables teams to move beyond simply creating cloud-first applications to building truly cloud-native ones.

Develop in the cloud, deploy to the cloud

An IDE that facilitates seamless development is now a standard expectation. However, the idea of portability within that environment is relatively novel, especially considering recent advancements in cloud-based IDEs. With the launch of GitHub Codespaces and the underlying DevContainers technology, developers are now able to develop code in a portable online environment. This setup allows them to utilize configuration files, enabling their development environment to be tailored to meet specific team requirements.

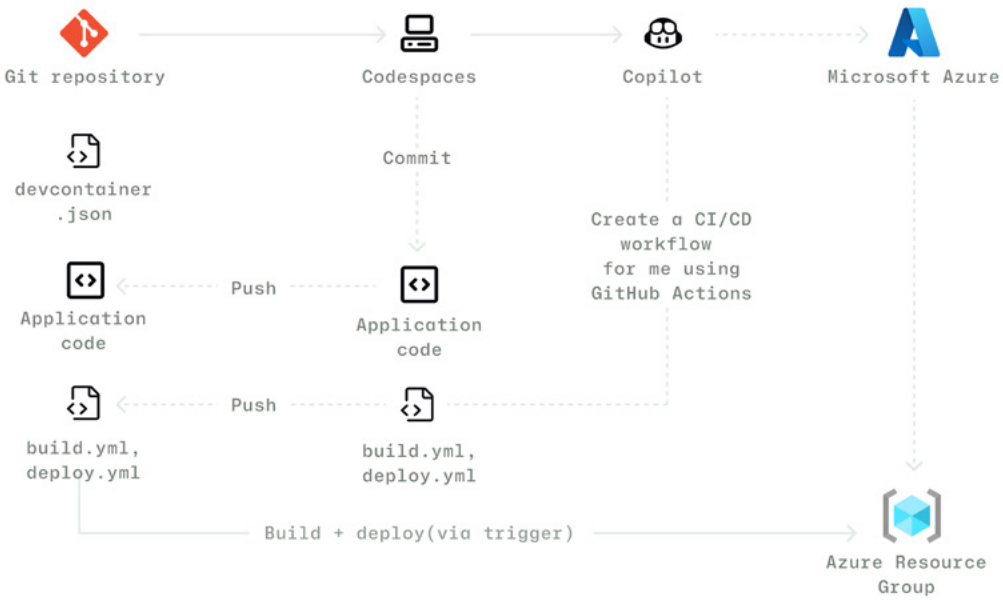


Figure 7: Portable online environment

The combination of reusability and portability offers organizations significant advantages. Teams can now centralize their configuration and environment specifications, enabling every developer—whether new or experienced—to work within the same setup. Having these centralized configurations allows team members to contribute to those configurations. As needs evolve, the environment can be updated and kept in a steady state for all developers.

Best of both worlds: GitHub Copilot for Azure

Since its launch, GitHub Copilot has served as an agentic AI assistant, helping developers tackle tasks ranging from the meaningful to the mundane. With the integration of Microsoft Azure knowledge bases, Copilot for Azure elevates this experience for development teams.

This powerful integration enables developers to request not only code examples, unit tests, and integration tests but also IaC scaffolding tailored to their current application. The knowledge base helps ensure that best practices are adhered to, allowing teams to verify compliance with standards such as the Cloud

Adoption Framework (CAF) and the Well-Architected Framework (WAF) for validating infrastructure and software designs.

To get started with GitHub Copilot for Azure, simply install the Copilot Chat extension in your preferred IDE and use the

@azure directive to customize your queries based on Azure-specific topics.

Managing developer processes and automation at scale

It's the developer workflow and time to market that really drive the metrics on productivity. Managing both the human processes that drive development and the automated workflows that power CI/CD can be a challenge, especially when many different teams are deploying to various clouds, cloud services, or on-premises environments.

Here are a few ways GitHub Enterprise takes the burden of managing workflows at scale:

- Simplify with reusable Actions and workflows
- Employ governance using Actions policies
- Use Actions published by verified publishers
- Use branch policies and rulesets to help ensure consistency and protect the mainline code
- Configure what makes sense at the enterprise and organization levels

End-to-end software lifecycle management

Managing both planned and in-flight work is an essential cornerstone of agile software development. GitHub Enterprise enables teams to create projects, link repositories, and track work items through issues, labels, and checklists—keeping progress visible and aligned with milestones. Labels can be used to differentiate between different types of issues.

For example, some of the default labels that can be used with issues are enhancement, bug, and feature. For any item that has an associated list of tasks related to the issue, it is possible to use Markdown to define that list of tasks as a checklist and include that in the body of the issue. This allows the tracking of completion based on that checklist and helps align it with project milestones, if defined.

Managing the feedback loop

It's no secret that the sooner a developer receives feedback about a specific functionality, the easier it is to fix potential issues and release updates compared to validating changes. Every organization has its own preferred method of communication, whether that's through instant messaging, email, comments on tickets or issues, or even phone calls. One additional GitHub Enterprise feature is Discussions, which offers developers and users the ability to interact in a forum-based environment, communicating changes, any types of issues with respect to functionality, or suggestions for new functionality that could then be translated into work items.

The feature set around Discussions has been popular with open source projects for quite some time. Some organizations may struggle to see the benefit of using Discussions when there are enterprise-level communication tools already in place. As organizations mature, being able to segregate communications that are relevant to specific software features and functionality, and then relaying those through Discussions that are associated with a specific repository, may give developers, product owners, and end users the ability to tightly interact in an environment that is specific to the features they are interested in seeing implemented.

Artifact lifecycles

Artifact management is one thing that is central to all software development lifecycles. Whether it's in the form of executables, binaries, dynamically linked libraries, static web code, or even through Docker container images or Helm charts, having a central place where all artifacts can be cataloged and retrieved for deployment is essential. GitHub Packages allows developers to store standardized package formats for distribution within an organization or an enterprise.

GitHub Packages supports the following:

- Maven
- Gradle
- npm
- Ruby
- .NET
- Docker images

Should you have artifacts that do not fall into those categories, you can still store them using the Releases feature in the repository. This allows you to attach required binaries or other files as needed.

Managing quality

Testing is an integral part of software development, whether that's executing unit or functional tests during a continuous integration build or having quality assurance analysts run through test scenarios to validate functionality within a web application. GitHub Actions allows you to integrate a variety of different testing types into your pipelines to help ensure that quality is being evaluated. In addition, GitHub Copilot can generate unit tests, taking the burden of creating unit tests and other test types off the developers and allowing teams to focus more on the business problem at hand.

Being able to easily integrate various testing utilities helps ensure quality is evaluated across the development lifecycle. As mentioned previously, you can use checks within GitHub Actions workflows to validate certain scenarios. This includes being able to successfully run a full suite of tests before allowing a request to be merged. Depending on the stage of deployment, you can also specify checks that include integration tests, load and stress tests, and even chaos tests to help ensure that applications going through the deployment pipeline are appropriately tested and validated before making it to production.

Conclusion

As you plan the next steps in your journey, it's important to think about continuing to bring benefits of AI and security to your DevOps process in order to deliver high-quality code that is secure from the start. By addressing productivity bottlenecks and eliminating time thieves, you can empower your engineers to work more efficiently. GitHub is ready to help you get started, no matter what solutions you're building or which phase of exploration you're in. Whether it's using GitHub Copilot to enhance the developer experience, safeguarding your security posture, or scaling with cloud-native development, GitHub is ready to help you every step of the way.

Next steps

→ To learn more about GitHub Enterprise or to start your free trial, visit https://gh.io/agentic-devops