WHITEPAPER

From Azure DevOps to GitHub

A practical guide to modernising your developer platform

Introduction

Software development is changing. AI is reshaping choices, not just code. In the past, developer choice meant selecting an IDE, language or framework. Today, AI is influencing not only how quickly code is written, but also which tools, platforms and workflows development teams use.

At the same time, development organisations are under pressure to deliver value faster without compromising security, quality or control.

Tool sprawl, disconnected workflows and repetitive manual tasks make it increasingly difficult.

For many organisations, Azure DevOps has been a reliable home for source control, build pipelines, work-item tracking and artefact management. But the tide has turned in software development.

GitHub Enterprise combines repositories, automation, security and AI-assisted development in a developer-first platform. The question is therefore not simply whether GitHub has more features.

The question is whether your current developer platform is ready for how your organisation wants to build software next.

Migrate to GitHub

“Move fast, stay secure. GitHub is where modern software and innovation happen. It brings developers, agents, and code together on one platform.”

Arian Geertsema, CTO Intercept

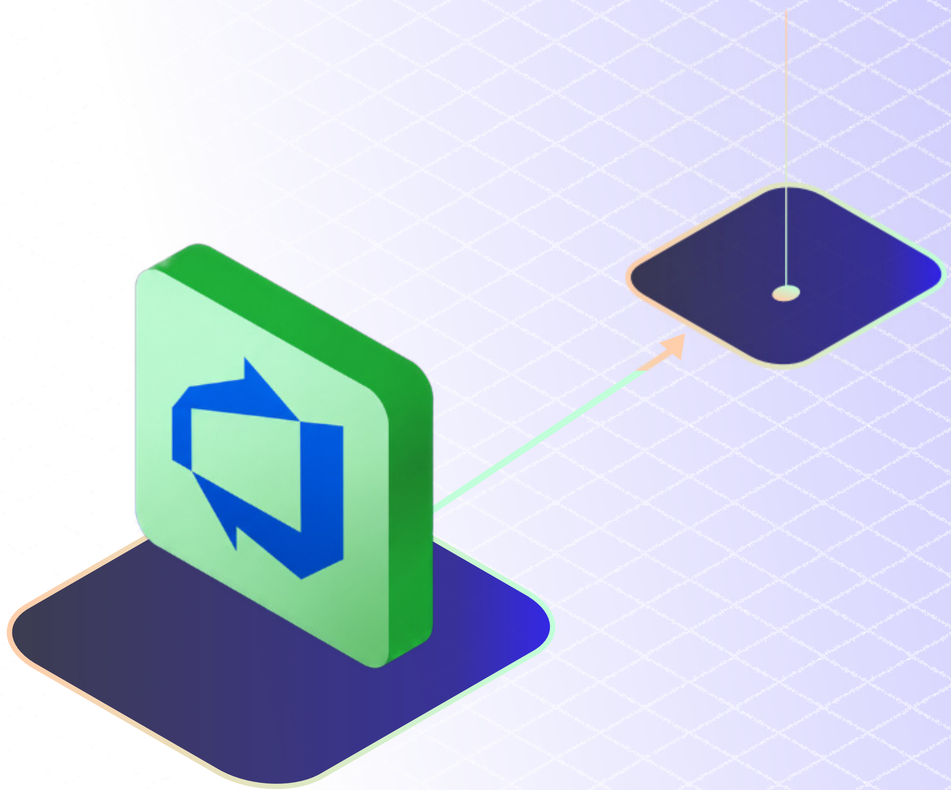
1. Azure DevOps has served organisations well

Azure DevOps is a cloud-based platform that provides integrated tools for software-development teams.

For years, Azure DevOps has been the default choice for many enterprise software teams. It manages source code, runs build pipelines, tracks work and houses artefacts under one roof.

But the dynamics are shifting.

Understand what Azure DevOps still does well, where GitHub Enterprise differs and why the platform decision is becoming more strategic.

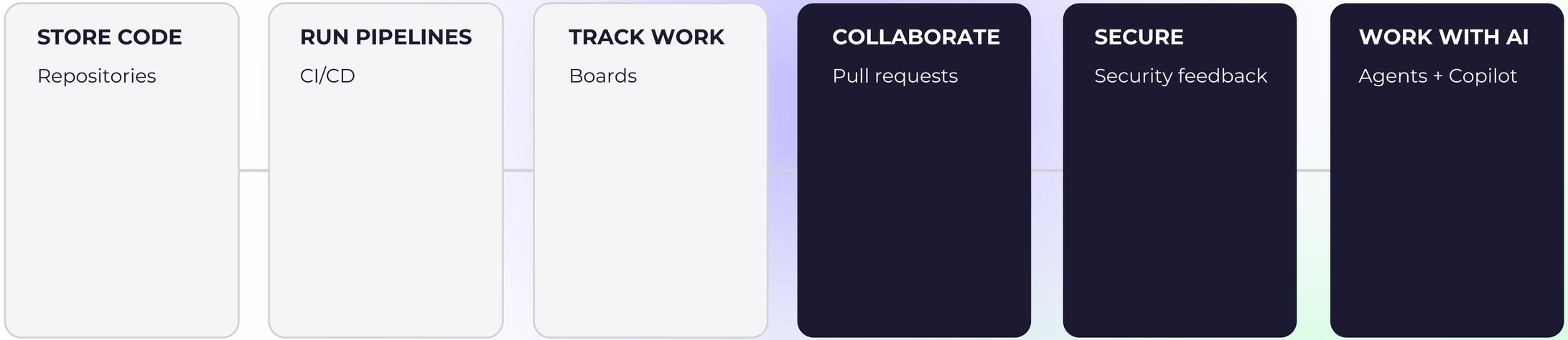


Azure DevOps does the job. But the dynamics are shifting.

The development platform is no longer only where code is stored and pipelines are run.

It is increasingly where developers collaborate, work with AI agents and receive security feedback while code is being written.

That is where GitHub Enterprise changes the conversation.



2. GitHub Enterprise: Unified platform built around developers

GitHub Enterprise is an enterprise-grade developer platform. It is available as a cloud-hosted service through GitHub Enterprise Cloud or as a self-hosted platform through GitHub Enterprise Server.

This gives organisations flexibility in managing infrastructure, data and compliance requirements.

It provides the core version control and collaboration features associated with GitHub, alongside enterprise capabilities for security, compliance, access control and AI-assisted development (GitHub Copilot).

One cohesive platform

GitHub Enterprise brings everything together in a single, cohesive experience:

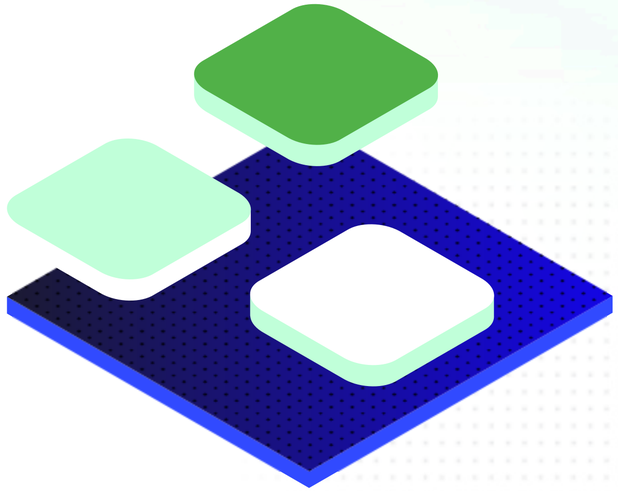
- **Repositories:**
  World-class Git hosting with code review, branch protection, and merge policies.

- **GitHub Actions:**
  A modern, YAML-based CI/CD platform with a massive marketplace of pre-built actions. It helps simplify code reviews, testing, and deployment, reducing manual tasks, shortening approval times, and accelerating development.

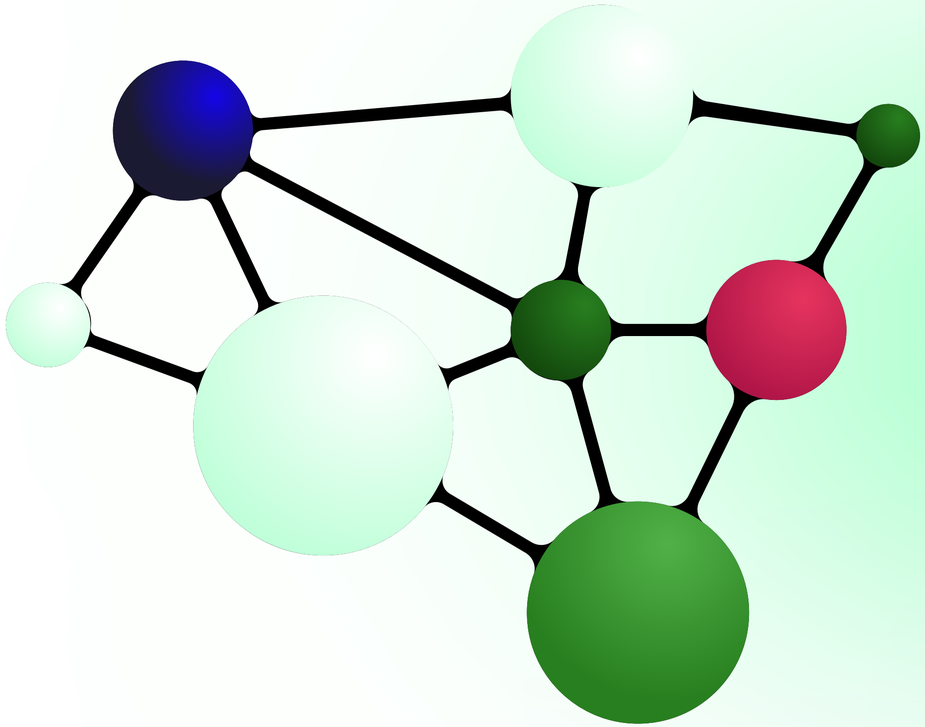
- **GitHub Advanced Security:**
  Code scanning, secret scanning and dependency management embedded in the development workflow.

- **GitHub Copilot:**
  AI assistance throughout the development process, from the IDE and command line to pull requests and coding agents.

- **GitHub Projects:**
  Flexible planning and work tracking integrated with issues, pull requests and repositories.



Together, these tools enable seamless collaboration, breaking down silos and allowing teams to manage every aspect of their projects efficiently—from planning to delivery.



3. The Network Effect of the World's Largest Developer Community

GitHub is home to more than 180 million developers and hosts more open-source projects than any other platform on the planet.

Every second, a new developer joins GitHub.

GitHub by the numbers

630M
TOTAL PROJECTS ON GITHUB

180M+ DEVS
MORE THAN 1 DEVELOPER JOINED GITHUB EVERY SECOND IN 2025

1.1B
TOTAL CONTRIBUTIONS TO PUBLIC PROJECTS ON GITHUB

4.3M
TOTAL AI PROJECTS

43.2M
PULL REQUESTS MERGED PER MONTH IN 2025 (UP 23% YOY)

TypeScript & Python
TOP TWO LANGUAGES USED IN 2025

Source: GitHub Octoverse 2025

GitHub: home for developers

GitHub started as a place developers actually wanted to be.

When your organisation moves to GitHub Enterprise, engineers can:

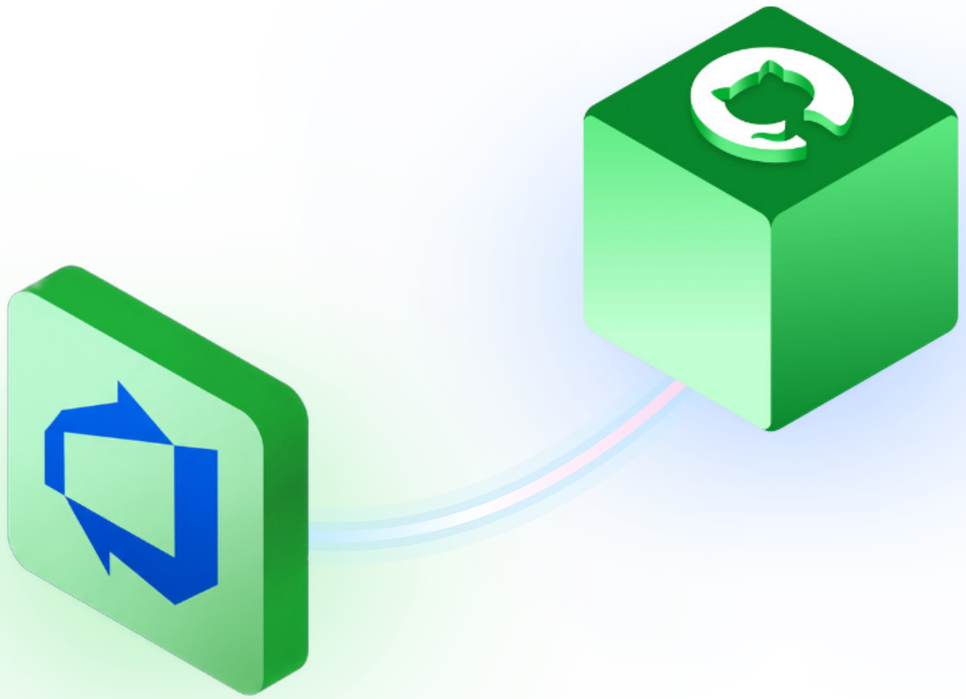
- Contribute to open-source projects without leaving the platform they use for work
- Discover and reuse community-maintained GitHub Actions
- Engage with the wider community around the libraries and frameworks they depend on
- Apply familiar GitHub workflows inside the organisation
- Attract talent that is already familiar and comfortable with GitHub

The pull request flow, repository structure, automation model and tone of the platform all point in the same direction. Developers like spending time there. That may sound soft until you try to hire engineers and realise they already have GitHub profiles. Their open-source work lives there, and so do the libraries, tools and frameworks your teams depend on.

4. Why move from Azure DevOps to GitHub Enterprise?

Moving from Azure DevOps to GitHub Enterprise is much more than a platform migration. It is a strategic investment in the developer experience, security posture and long-term agility of your engineering organisation.

For organisations still working primarily in Azure DevOps, six reasons make the case for moving towards GitHub Enterprise.



A unified platform built around developers

GitHub was built from the ground up as a developer-centric environment. Every feature, workflow, and integration has been designed with the developer experience at its core.

Rather than just a code-hosting platform, it offers a complete developer experience that combines collaboration, automation, security, and AI-assisted development in ways other platforms like Azure DevOps simply cannot match. This reduces silos and allows teams to manage development from planning through delivery without constantly moving between separate tools and workflows.

Comparison

Azure DevOps, by contrast, evolved from Team Foundation Server (TFS), a tool originally built around project management and process governance. Whilst Microsoft has done significant work to modernise Azure DevOps, the legacy of its origins is still visible in the UI, the configuration model, and the way teams interact with it day-to-day.

Repository Management: A Step Forward

GitHub started as a place developers actually wanted to be.

Pull Requests and Code Review

GitHub pull requests are the gold standard for code review. Features such as suggested changes, code owners, required reviews, and inline comments are mature and well understood across the industry. Azure DevOps pull requests work, but the experience is less polished and less widely adopted.

Branch Protection Rules

GitHub's branch protection model is highly configurable. You can require status checks, enforce linear history, restrict who can push to protected branches, and require signed commits, all from a simple UI or via code using the GitHub API.

Repository Templates & Standardisation

GitHub repository templates allow teams to bootstrap new projects with a consistent structure, including default workflows, code owners files, issue templates, and contributing guides. This makes standardisation across an engineering organisation much easier.

Innersource and discoverability

GitHub Enterprise supports internal repositories that are visible to all members of the organisation without being public. This encourages an innersource culture where teams can discover and reuse each other's code, just as they would with open source.

CI/CD with GitHub Actions: Modern by Design

While most DevOps tools provide CI/CD capabilities, GitHub goes a step further with GitHub Actions. It is a fully cloud-native, event-driven automation platform that supports enterprise-grade sprovioftware delivery across cloud, on-premises and hybrid environments.

With GitHub Actions, teams can host their CI/CD pipelines and automate activity throughout the development workflow. Actions can be triggered by virtually any event in the GitHub ecosystem, including:

- **A push**
- **A pull request**
- **A release**
- **A schedule**
- **A manual workflow dispatch**
- **A comment on an issue**
- **A security event**

Azure Pipelines vs GitHub Actions

Where Azure Pipelines uses a YAML format that can feel verbose and complex, GitHub Actions is structured around simple, composable jobs and steps. The marketplace contains tens of thousands of pre-built Actions for everything from deploying to Azure, AWS, or GCP, to sending Slack notifications, publishing npm packages, or running security scans.

This event-driven model makes it easier to build sophisticated automation without treating every workflow as a traditional build or deployment pipeline.

Rethinking the CI/CD era

GitHub also supports a broad ecosystem of integrations, allowing teams to connect the CI/CD tools and services they already use.

This can reduce tooling fragmentation, streamline workflows and improve productivity.

- **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 organisational standards.
- **Increased efficiency:** Automate repetitive tasks to free up developers' time.

For teams already on Azure
GitHub Actions has first-class support for deploying to Azure services, with official Microsoft-maintained Actions for Azure App Service, Azure Container Apps, Azure Kubernetes Service, and more.

GitHub Advanced Security: Shifting Security Left

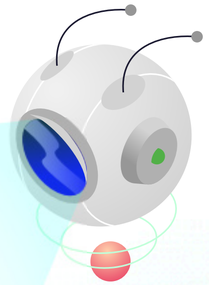
In most Azure DevOps shops, security still works the old way: a scan before release, a pen test before go-live, and findings dumped on developers weeks after the code was written.

GitHub Enterprise includes **GitHub Advanced Security** (GHAS), a comprehensive suite of tools that integrates security directly into the developer workflow, rather than treating security as a gate at the end of the SDLC.

Security is embedded throughout the process, from dependency management and vulnerability detection to preventive controls that protect sensitive information. This helps strengthen the organisation's software security posture while allowing developers to stay focused, reduce context switching and maintain their coding flow.

Security as a first-class citizen

GitHub Advanced Security is built around three core capabilities:



CODE SCANNING WITH CODEQL

CodeQL is GitHub's code analysis engine and is the default scanner behind GitHub's code scanning feature. It treats code as data - compiling it into a queryable database and then running semantic analysis queries against that database to find vulnerabilities.

CodeQL supports languages including JavaScript, TypeScript, Python, Java, C#, Go, Ruby, C and C++. For many languages, enabling code scanning is as straightforward as adding the official CodeQL workflow to the repository.

SECRET SCANNING

Secret scanning automatically detects API keys, tokens, connection strings, private keys and other credentials committed to a repository. It can analyse both the current repository and its complete commit history.

Push protection goes one step further by blocking supported credentials before they are pushed. With a low false-positive rate and approximately 150 service-provider integrations, it can support faster credential revocation and rotation while reducing disruption for developers.

DEPENDENCY REVIEW: DEPENDABOT

Modern applications depend on dozens of referenced packages, each of which may introduce many additional dependencies. Across hundreds of repositories, understanding which dependencies are in use, which contain known vulnerabilities and which licences apply can become difficult.

GitHub Enterprise immediate insight into dependency graphs. Dependency review helps identify risk before new packages are introduced, while Dependabot alerts flag outdated or vulnerable libraries that may require remediation.

***Azure DevOps** supports third-party security integrations, but these often require additional configuration, licensing and pipeline work. In GitHub Enterprise, the core capabilities are built directly into the platform and surfaced in the same context where developers already work.*

Total cost of ownership and consolidation

CURRENT TOOLSPRAWL

Many organisations that run Azure DevOps also have GitHub accounts for open-source work, third-party integrations or community engagement.

They may also rely on separate tools for security scanning, dependency management, CI/CD, repository governance and AI-assisted development. Each additional platform introduces more licences, administration, integrations and places where teams need to search for information.

CONSOLIDATED PLATFORM

Consolidating on GitHub Enterprise eliminates tooling duplication, reduces the administrative overhead of managing two platforms, and provides a single pane of glass for all development activity. This can lead to less context switching, fewer disconnected workflows and less time spent moving between tools.

75% of developers lose between 6 and 15 hours per week to tool sprawl.

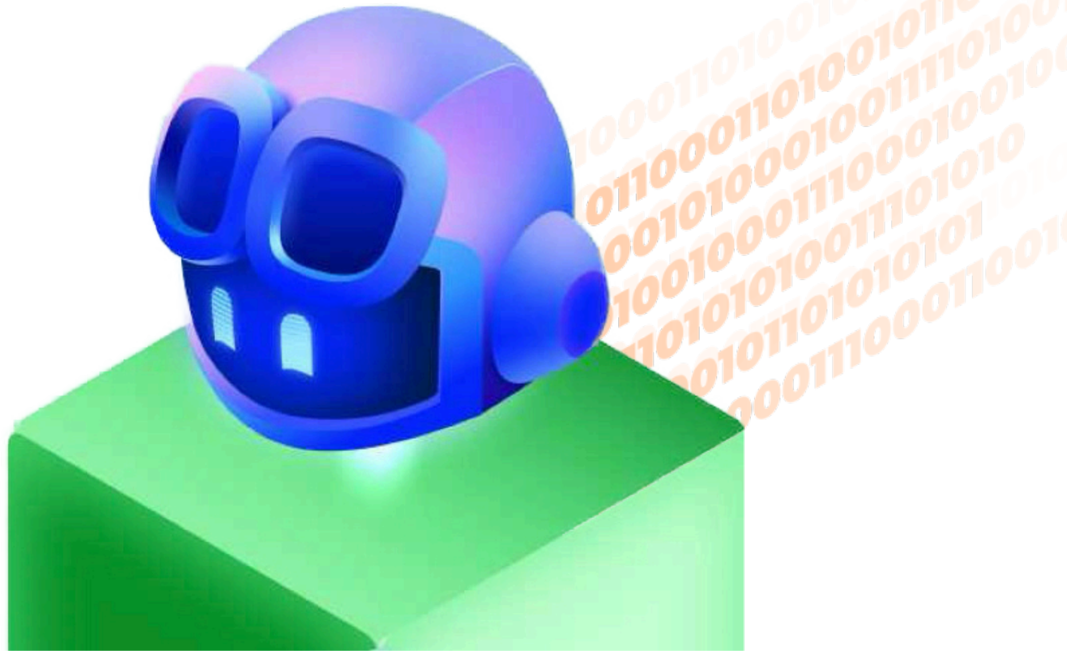
AI-powered development with GitHub Copilot

Perhaps the most transformative capability available in GitHub Enterprise is GitHub Copilot. Copilot is an AI pair programmer that integrates directly into Visual Studio Code, JetBrains IDEs, Visual Studio, and more. It suggests code completions, generates entire functions, writes tests, and even explains unfamiliar code.

For enterprise teams, GitHub Copilot can use context from the organisation's repositories, internal libraries and coding conventions to provide more relevant suggestions that reflect how the team works.

GitHub is where Microsoft's investment in AI-powered software development is most visible.

The depth of integration between GitHub Copilot and the GitHub development workflow is unmatched. For organisations considering a move from Azure DevOps, that matters.



5. GitHub Copilot and AI-powered development

GitHub Copilot is the most widely adopted AI coding assistant in the world. It is deeply integrated into GitHub Enterprise and into the editors, terminals, and browser interfaces that developers use every day.

Unlike productivity tools that sit adjacent to the development workflow, Copilot is woven into it, generating code, reviewing pull requests, explaining unfamiliar codebases, writing tests, and now operating as an autonomous agent capable of implementing entire features.

This represents a global shift in the way software is developed.

What GitHub Copilot can deliver

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

15% higher code quality with Copilot

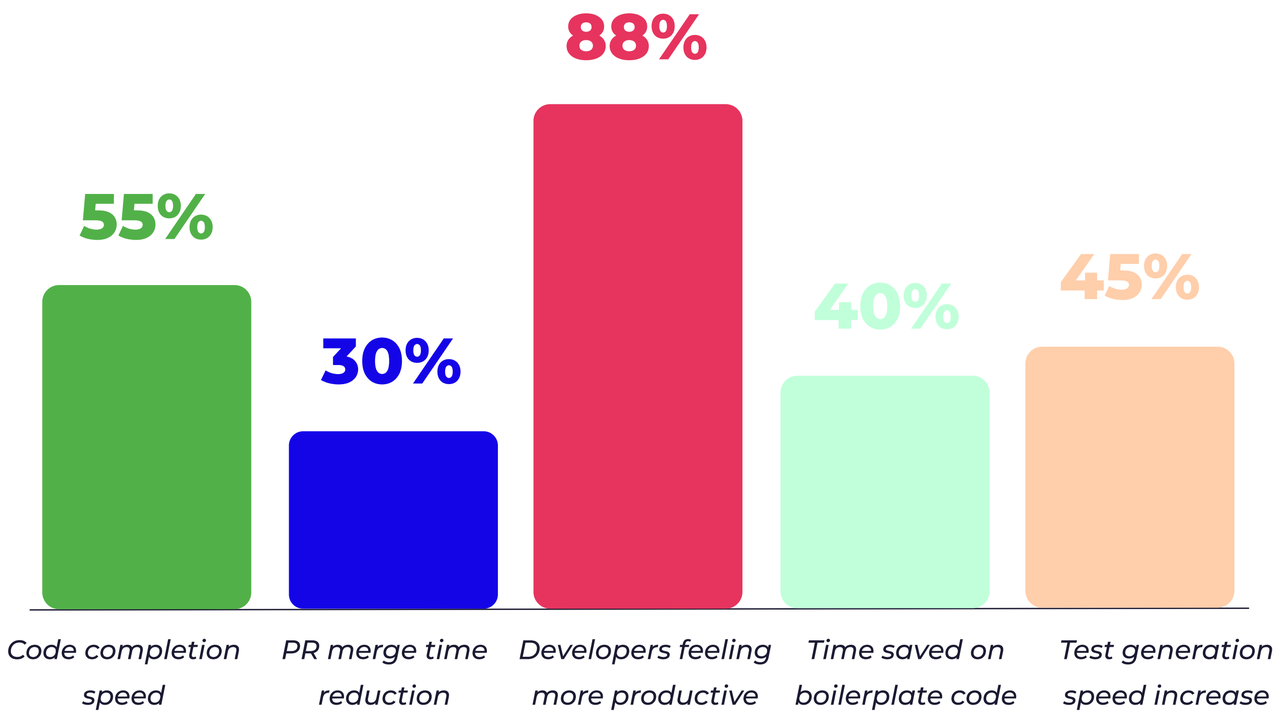
These benefits can translate into measurable outcomes for your organisation, such as:

- **Increased efficiency:** Automate repetitive tasks, minimise 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:** Utilise Copilot to support strategic goals, improve customer experiences, and maintain a competitive edge in the market.
- **Reduce repetitive tasks:** 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.

Measuring the Impact: What the Research Shows

GitHub has published data from large-scale studies of Copilot usage that provide a basis for estimating the productivity impact for your organisation.



60%

Developers maintain flow state longer

3.5x

Faster onboarding for new codebases

60%

Reduction in context switching

The GitHub Copilot Product Suite

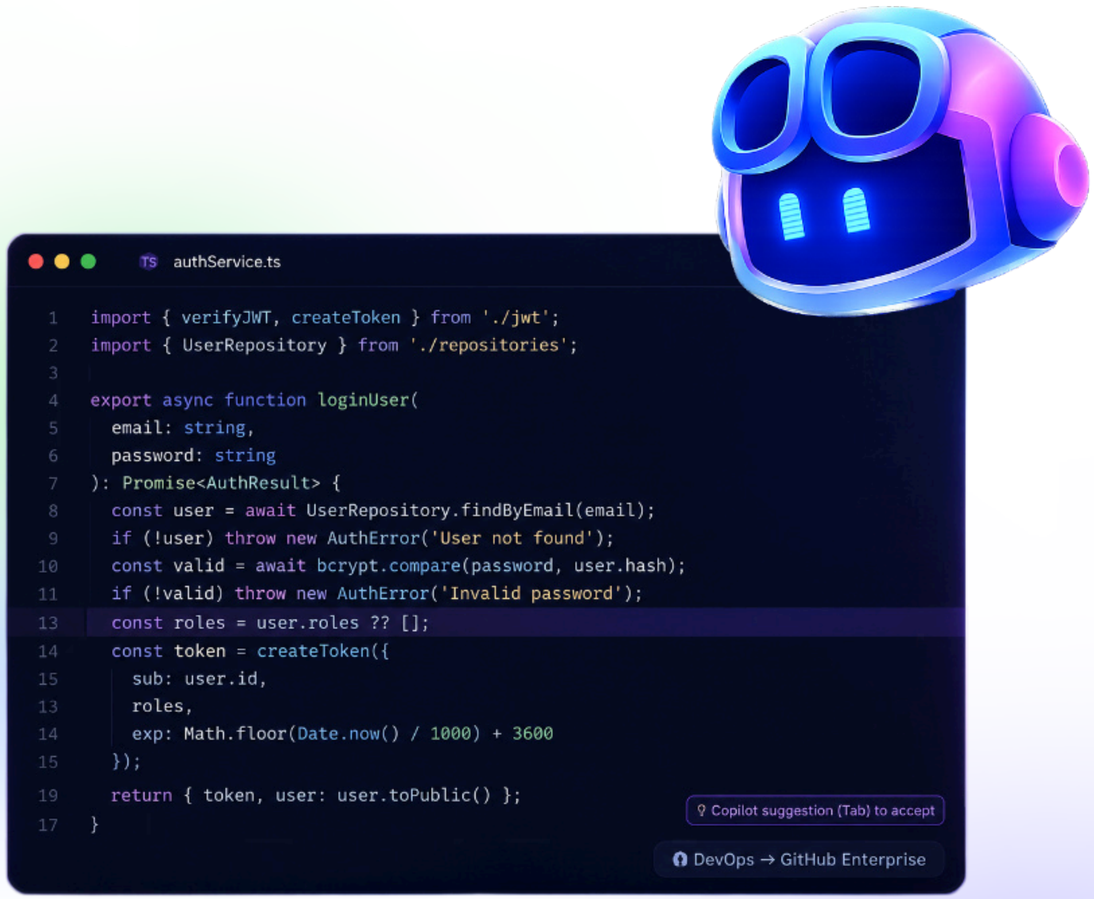
GitHub Copilot in the IDE

The original Copilot experience provides real-time code completions in:

- Visual Studio Code
- Visual Studio
- JetBrains IDEs
- Vim
- Neovim
- Xcode
- Eclipse

Completions are triggered as a developer types and range from single-line suggestions to entire function implementations. The underlying model has access to the file being edited, surrounding open files, and, in the Enterprise tier, an index of the organisation's own repositories.

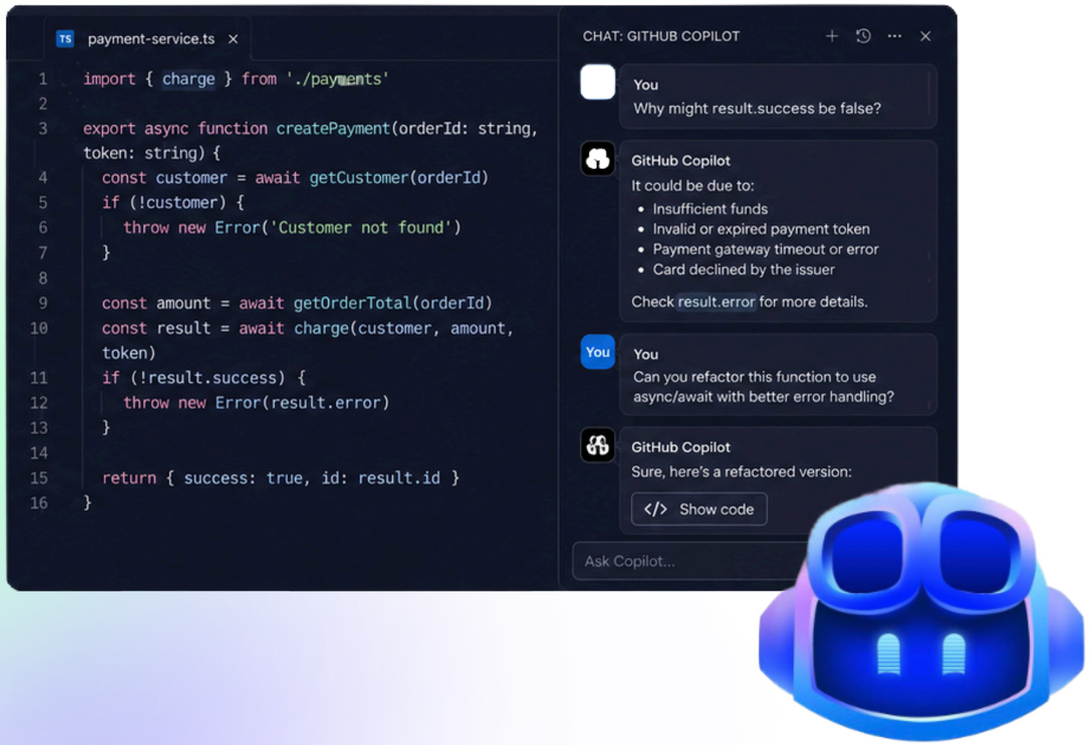
This allows Copilot to produce suggestions that reflect not just general programming patterns, but the specific conventions, libraries, and idioms used by your team.



GitHub Copilot Chat

Copilot Chat is a conversational AI interface embedded in the IDE and on GitHub.com.

It allows developers to:



- → **Ask questions about the codebase in natural language:**
  "What does this function return if the input is null?"
- → **Request explanations of code they did not write:**
  "Explain what this regex matches"
- → **Get help debugging:**
  "Why is this test failing? Here is the error output"
- → **Ask for refactoring suggestions:**
  "Rewrite this method to use async/await instead of callbacks"
- → **Generate code from a description:**
  "Write a GitHub Actions workflow that builds a Docker image and pushes it to GitHub Container Registry"

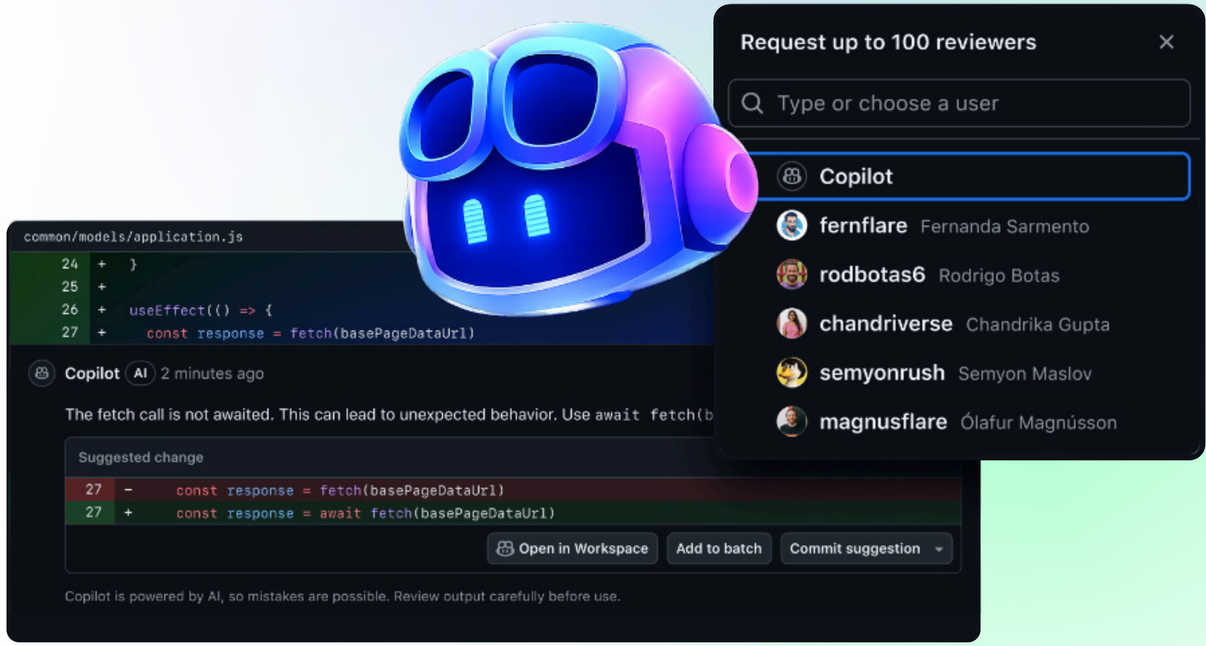
GitHub Copilot for Pull Requests

Copilot integrates directly into the GitHub pull request review process in several ways:

→ **Copilot Code Review:**
When a developer opens a pull request, they can request a Copilot code review. Copilot analyses the diff and produces inline comments that identify potential issues, suggest improvements, and flag code that does not follow established patterns in the codebase. These comments are indistinguishable from those of human reviewers and can be accepted with a single click.

→ **Pull Request Summaries:**
Copilot can automatically generate a summary of what a pull request does, what files it touches, and what the reviewer should pay particular attention to. This dramatically reduces the cognitive load of reviewing large pull requests and is especially valuable when the author and reviewer are in different time zones.

→ **Suggested changes:**
Copilot can generate code suggestions as part of its review that the pull request author can accept, reject, or modify directly in the GitHub UI.

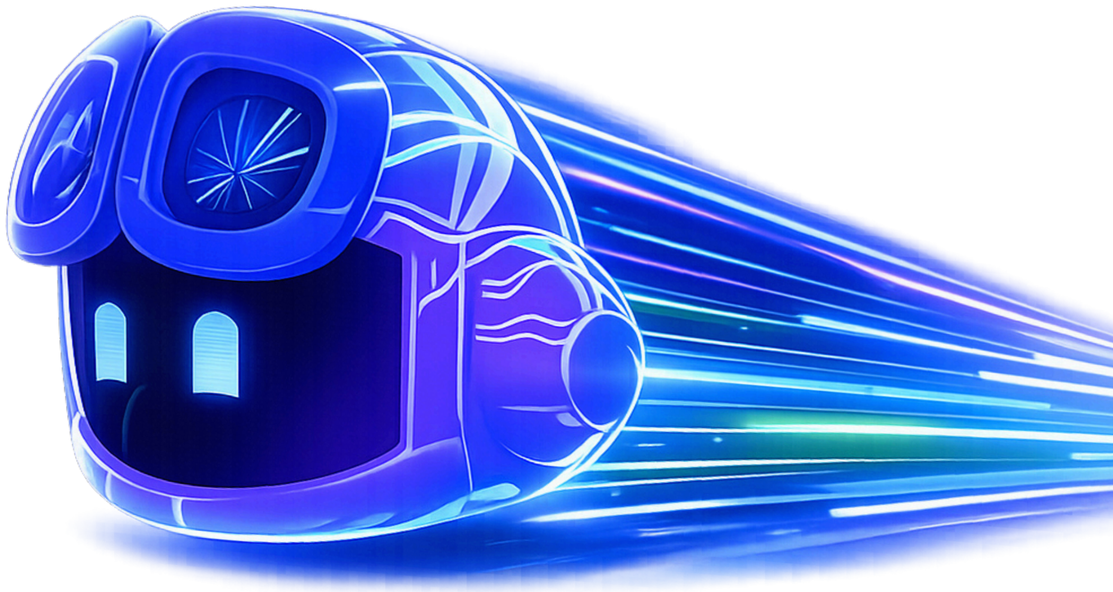


GitHub Copilot Workspace (Agentic Development)

Agentic development represents the leading edge of GitHub's AI investment. Copilot no longer only assists with individual lines or functions. It can take on complete development tasks.

A developer can describe a task such as: *"Add pagination to the user's endpoint in the API" and Copilot Workspace:*

- → **Reads** the relevant code and **understands** the existing patterns
- → **Proposes a plan:** which files need to change and what each change will do
- → **Implements the changes** across multiple files
- → **Writes** or **updates** test
- → **Proposes** a pull request description



Microsoft Copilot vs GitHub Copilot in Azure DevOps

Does Azure DevOps offer anything comparable?

Selected capabilities in Azure DevOps

Microsoft has introduced Copilot features around work-item summarisation, pipeline assistance and pull-request descriptions in Azure DevOps.

The fundamental difference

However, these are surface-level integrations that do not approach the depth of GitHub Copilot. The **fundamental difference is that GitHub Copilot has been developed as a deep, native part** of the GitHub platform since 2021. Every aspect of the product completions, chat, PR review, workspace, and extensions is built with the developer workflow in mind and improves with each iteration. The Azure DevOps Copilot integrations are additions to a platform whose core architecture was not designed for them.

Where Microsoft’s AI investment is concentrated

For organisations evaluating the migration and asking where the AI capability sits, the answer is clear: GitHub Enterprise, not Azure DevOps, is where Microsoft's AI investment in developer tooling is concentrated. And the gap will only keep growing.

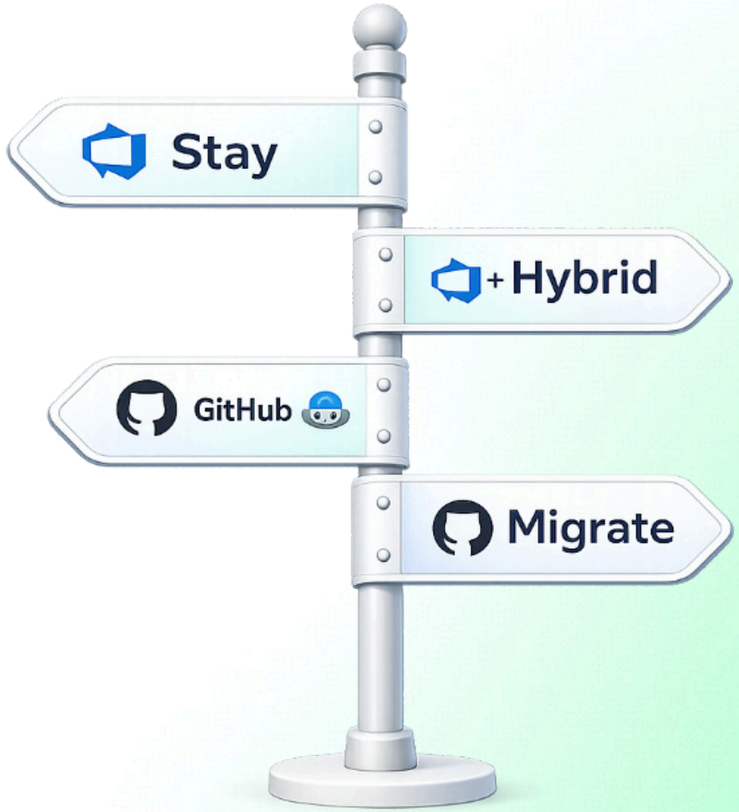
6. Stay, hybrid or migrate?

A move to GitHub Enterprise does not have to start with a full replacement of Azure DevOps.

Organisations can adopt GitHub in stages, depending on where they expect the greatest value and which Azure DevOps components still support their current way of working. The migration itself is well documented and, by now, well travelled by organisations that have already made the move.

The return can show up in a stronger security posture, better pipeline performance, an improved developer experience and the productivity gains created by GitHub Copilot.

Let’s look at the different routes.



Choose the path that fits your organisation

1. Azure DevOps with GitHub Copilot in the IDE

Teams remain on Azure DevOps but begin using GitHub Copilot in their existing IDEs. Developers gain AI-assisted coding, test generation, code explanation and refactoring without changing repositories, pipelines or work tracking.

Key note	This only captures part of the value. The benefits of GitHub around repositories, pull requests, Advanced Security and agentic development remain limited while the development platform stays centred on Azure DevOps.

2. Azure DevOps with selected GitHub capabilities

Organisations introduce selected GitHub capabilities alongside Azure DevOps. Possible starting points include one development team, new repositories, GitHub Copilot, GitHub Advanced Security, pull-request collaboration or selected GitHub Actions workflows. Azure Boards, Azure Pipelines, Azure Test Plans and legacy workflows can remain where needed.

Key note	Retaining Azure DevOps should be an active decision. Be clear about which components remain, why they remain and whether the setup is transitional or part of the long-term platform model.

Choose the path that fits your organisation

3. Github repositories with Azure DevOps services

Source code, pull requests and code review move to GitHub, while selected services such as Azure Boards, Azure Pipelines and Azure Test Plans remain in place. This gives developers access to GitHub's repository experience, AI assistance and integrated security while preserving established planning, testing or deployment processes. It can be useful when Azure Boards workflows or pipelines are too complex or business-critical to migrate immediately.

Key note	Moving repositories shifts the centre of the developer workflow to GitHub and creates a foundation for migrating security, automation and other platform capabilities later.

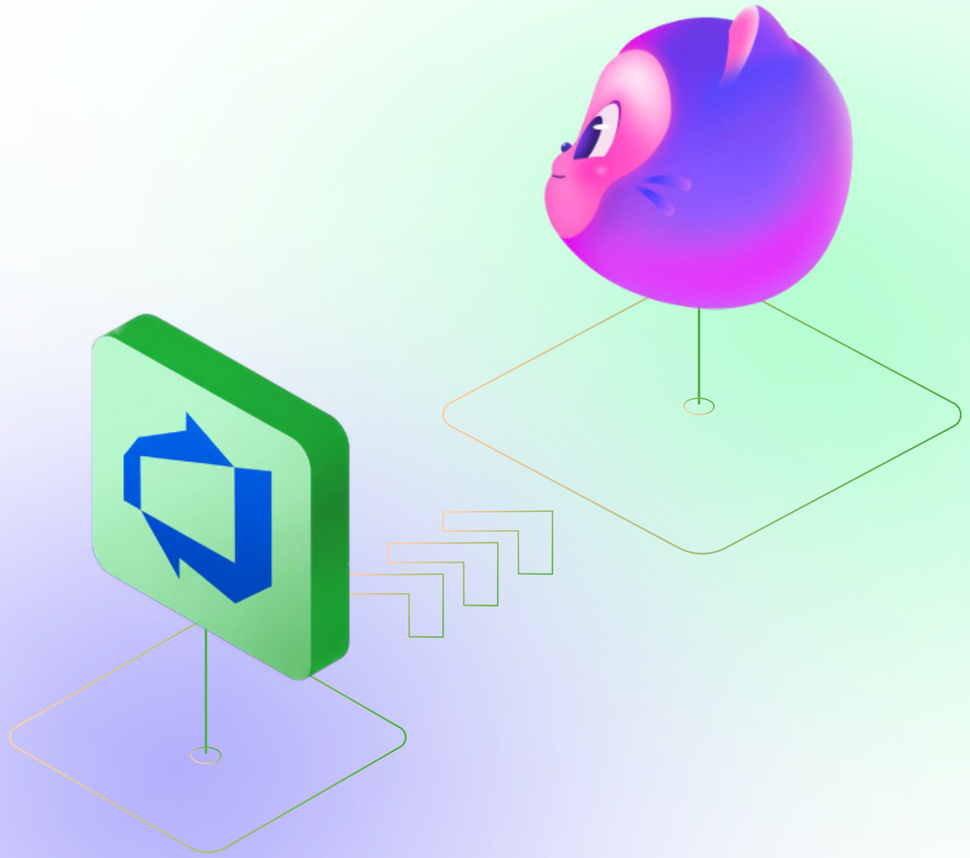
4. Full migration to GitHub

A full migration moves both the development workflow and the supporting platform capabilities to GitHub. Repositories, pipelines, service connections, permissions, work items and external integrations all need to be assessed. For most organisations, this is best approached progressively. Start with a pilot team or limited repository set, validate the approach and migrate additional teams in planned waves.

Key note	A full migration offers the strongest consolidation benefits, but also requires the most preparation. Azure DevOps components can be retired gradually once they are no longer required.

7. Planning your move to GitHub Enterprise

Once you have decided how far you want to move towards GitHub, the next step is to define what to move, how the target environment should be structured, and how to complete the migration without disrupting development.



Define what needs to move

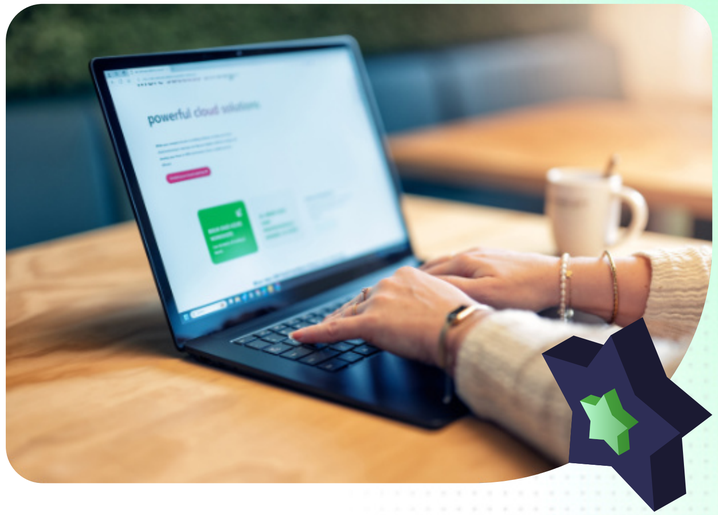
Start by taking inventory of the Azure DevOps environment:

- **Repositories and their history**
- **CI/CD pipelines**
- **Work items**
- **Wikis**
- **Packages and artefacts**
- **Test plans**
- **Permissions**
- **Service connections**
- **Webhooks and external integrations**
- **Security tooling**

Migrating everything by default can carry obsolete repositories, processes and dependencies into the new platform.

Carefully review your Azure DevOps projects and repositories to determine what is essential to move to GitHub. Migrate only what is active, required or needed for compliance, and archive obsolete repositories, work items, wikis and packages in Azure DevOps.

This way, you'll have a clean and well-organised GitHub environment.



For work items, focus on active items such as New or In Progress work. Historical items can remain in Azure DevOps unless they are needed for compliance or reference. Audit packages in the same way and move only the versions still in use.

Design the target GitHub environment

Before moving repositories, define how GitHub Enterprise should be structured and governed.

For many companies, one GitHub organisation is sufficient. Multiple organisations may be needed where separate access controls, policies or spending limits apply. However, no single structure fits every organisation.

- The setup should reflect your security, ownership and governance requirements.
- You should also design the team and permissions structure before migration.

Teams and permissions are not automatically migrated by GitHub Enterprise Importer, so they need to be recreated in GitHub. Define these together with repository ownership, naming standards, branch protection, CODEOWNERS, security policies and reusable workflows. This creates a consistent setup that can be applied across migration waves and used for both migrated and new repositories.

Prepare repository migration

For many organisations, repository migration is the first major practical step. Doing it well sets the tone for everything that follows.

Repositories contain more than source code. They can include:

- Repositories and their history
- CI/CD pipelines
- Work items
- Wikis
- Packages and artefacts
- Test plans
- Permissions
- Service connections
- Webhooks and external integrations
- Security tooling

Before migrating, decide which of these elements need to be preserved. Full Git history is usually worth keeping, but pull-request history may not be necessary unless it is required for compliance, auditing or future reference.

There are two main migration approaches:

Manual Git migration
Preserves commit history, but not pull requests or branch policies.

GitHub Enterprise Importer
Migrates Git history, pull requests and selected settings, with support for bulk migrations.

Tips for a smooth migration

Audit repositories:
Review activity, size and ownership before grouping repositories into migration waves.

Plan the cutover:
Set a clear cutover date and make the Azure DevOps repository read- only during the move.

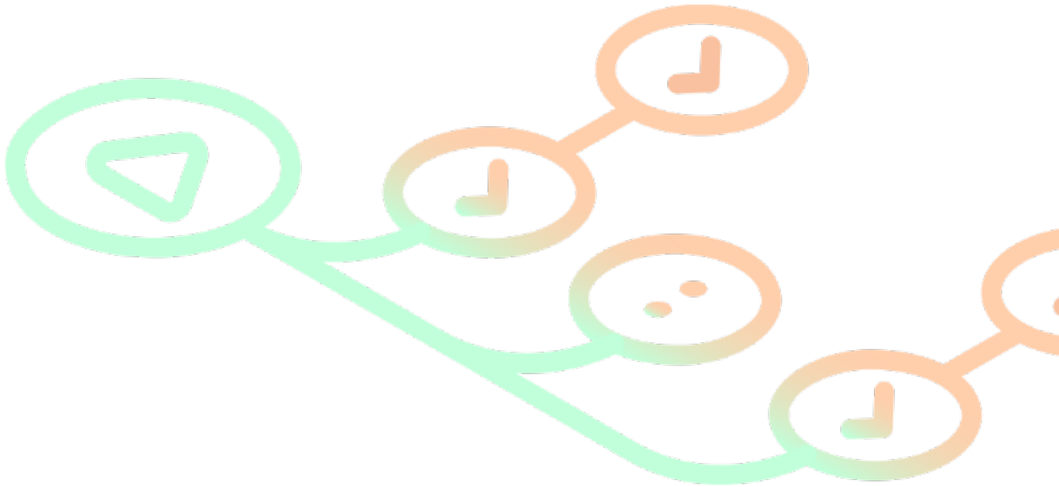
Handle wikis with care:
Account for the structural differences between Azure DevOps and GitHub, as wiki migration may require manual work.

Organise shared content:
For projects spanning multiple repositories, consider a dedicated GitHub repository for issues and wiki content.

Update integrations:
Redirect service connections, webhooks, pipelines, deployment tools and repository URLs.

Validate the migration:
Check branches, tags, history, builds and integrations before archiving the Azure DevOps repository.

Migrating pipelines



Pipeline migration is often the most technically involved part of the move and requires more thought than a simple lift-and-shift.

Azure Pipelines and GitHub Actions share concepts, but differ in their syntax, execution and runner models.

Start by reviewing

- → Pipeline triggers, stages, jobs and tasks
- → Variables, secrets and approvals
- → Service connections and runners
- → Reusable templates and deployment dependencies
- → Network and artefact requirements

Note:
Not every Azure Pipeline needs to move immediately. Business-critical or highly customised pipelines can remain temporarily, while newer or less complex pipelines may provide a better starting point.

Tips for a smooth migration

- **Use maintained Actions:**

  Use the GitHub Actions Marketplace for common build, test and deployment tasks.

- **Centralise reusable workflows:**

  Make shared workflows available across repositories at the organisation level.

- **Audit service connections:**

  Replace Azure DevOps service connections with GitHub Secrets or OIDC federated credentials where appropriate.

- **Migrate incrementally:**

  Run the GitHub Actions workflow alongside the existing Azure Pipeline before cutover to compare outputs and resolve issues without interrupting delivery.

Pilot, validate and migrate in waves

Start with a low-risk pilot. Validate and refine the approach before moving business-critical repositories.

Migrate in waves

Start with the repositories no one will cry over if something goes sideways.

Validate the migration, update integrations and move to business-critical repositories once you trust the process.

Validate approach

Test permissions, repository settings, builds, deployments and integrations.

Document issues and refine the migration approach before the next wave.

Apply GitHub standards

Once a repository lands on GitHub, use branch protection, CODEOWNERS and repository templates to create a more consistent governance model.

Scale with confidence

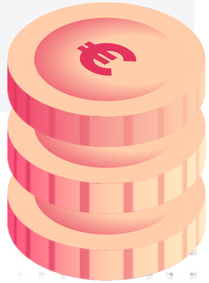
Consolidating on GitHub Enterprise eliminates tooling duplication, reduces the administrative overhead of manants.

The cost of doing nothing

TODAY

Many organisations that run Azure DevOps also have GitHub accounts for open-source work, third-party integrations or community engagement.

They may also rely on separate tools for security scanning, dependency management, CI/CD, repository governance and AI-assisted development. Each additional platform introduces more licences, administration, integrations and places where teams need to search for information.



THE GAP WIDENS

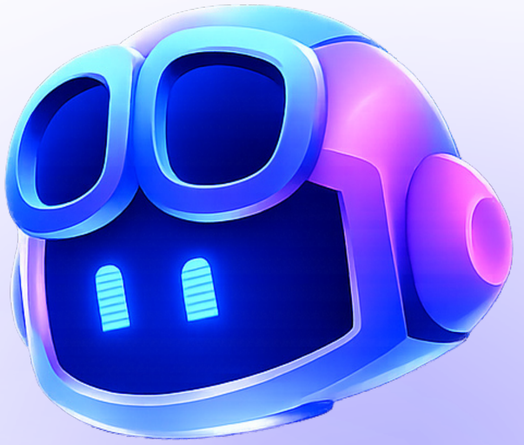
Consolidating on GitHub Enterprise eliminates tooling duplication, reduces the administrative overhead of managing two platforms, and provides a single pane of glass for all development activity. This can lead to less context switching, fewer disconnected workflows and less time spent moving between tools.

Bottom line

Moving from Azure DevOps to GitHub Enterprise is not just a platform migration; it is a strategic investment in the developer experience, the security posture, and the long-term agility of your engineering organisation. GitHub Enterprise offers a richer, more integrated, and more modern environment for building software, with capabilities in AI assistance, security automation, and community collaboration that Azure DevOps cannot match.

The aggregate of these advantages, include:

- Better developer experience
- Stronger security posture
- Better pipeline performance
- Less tool sprawl
- Faster delivery and AI assistance embedded throughout the workflow



GitHub is Where Modern Software Happens

Organisations who migrate to GitHub position themselves to build better software, faster and more securely.

The ROI is measurable, and the cost of delay is real: every month spent on Azure DevOps is a month where your engineering organisation is not benefiting from the capabilities that GitHub Enterprise offers today. At the current pace of innovation, that gap will only widen.

Ready to move to GitHub?

Establish a secure and governed GitHub foundation, migrate repositories and CI/CD pipelines, and help teams adopt GitHub Copilot and GitHub Advanced Security without disrupting ongoing development.

We ensure a production-ready GitHub platform, a controlled migration path and a development environment built for AI, security and faster software delivery.

Migrate to GitHub with Intercept →

contact@intercept

About Intercept: Your Azure & GitHub partner

Intercept is one of only three companies in the Benelux that combines the Microsoft Specialization: Agentic DevOps with Microsoft Azure and GitHub with Azure Expert MSP status. This recognises our ability to support organisations across assessment, design, pilot, implementation and optimisation.

At Intercept we help your organisation move to GitHub through a phased or full migration, depending on your team and platform requirements, and existing Azure DevOps investments.

Microsoft Solutions Partner

ISO 27001 en ISO 9001 gecertificeerd

MVP Microsoft Most Valuable Professional

Microsoft Partner | Azure Expert MSP

Microsoft Solutions Partner
Digital & App Innovation
Azure
Specialist
Agentic DevOps with Microsoft Azure and GitHub

CSA Star L1 level 1 certificaat

Western Europe Partner Pledge
Microsoft

We are happy to assist
Please contact us if you have any questions after reading or want more information on a specific topic. We are happy to help.

contact@intercept
+31 (0) 38 77 79 820