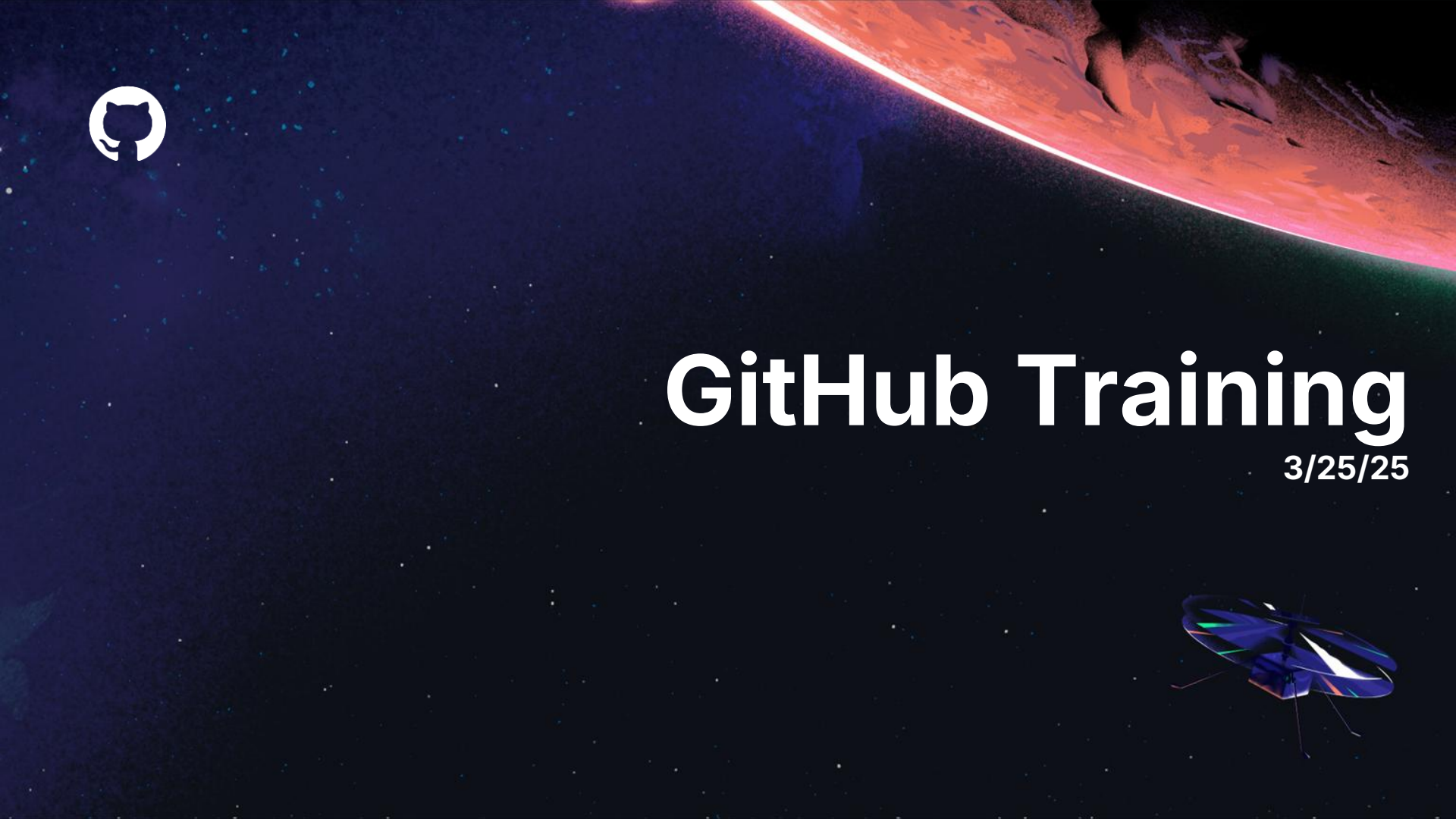

GitHub Training
3/25/25

AGENDA

GitHub Codespaces

GitHub Actions

GitHub Advanced Security

GitHub Copilot

GitHub Platform

Copilot

Collaboration

- Pull requests
- Projects
- Issues
- Discussions
- Merge queue
- Search

Productivity

- CI / CD with Actions
- Automation
- Copilot
- Codespaces
- Runners
- npm
- Mobile

Security

- Advanced Security
- Secret scanning
- Code scanning
- Security overview
- Supply chain
- Dependabot

Scale

- 100M+ developers
- Source code management

GitHub Codespaces

Modern cloud development

Improving your organization’s software development process with Codespaces

Impact of local development and expanding engineering teams

Onboarding friction

New engineers spending more time figuring out how to set up locally rather than understanding the codebase.

Over or under investment in developer experience

Daily tools for your engineering teams should *just work.*

Opportunities

Improving engineer satisfaction and productivity

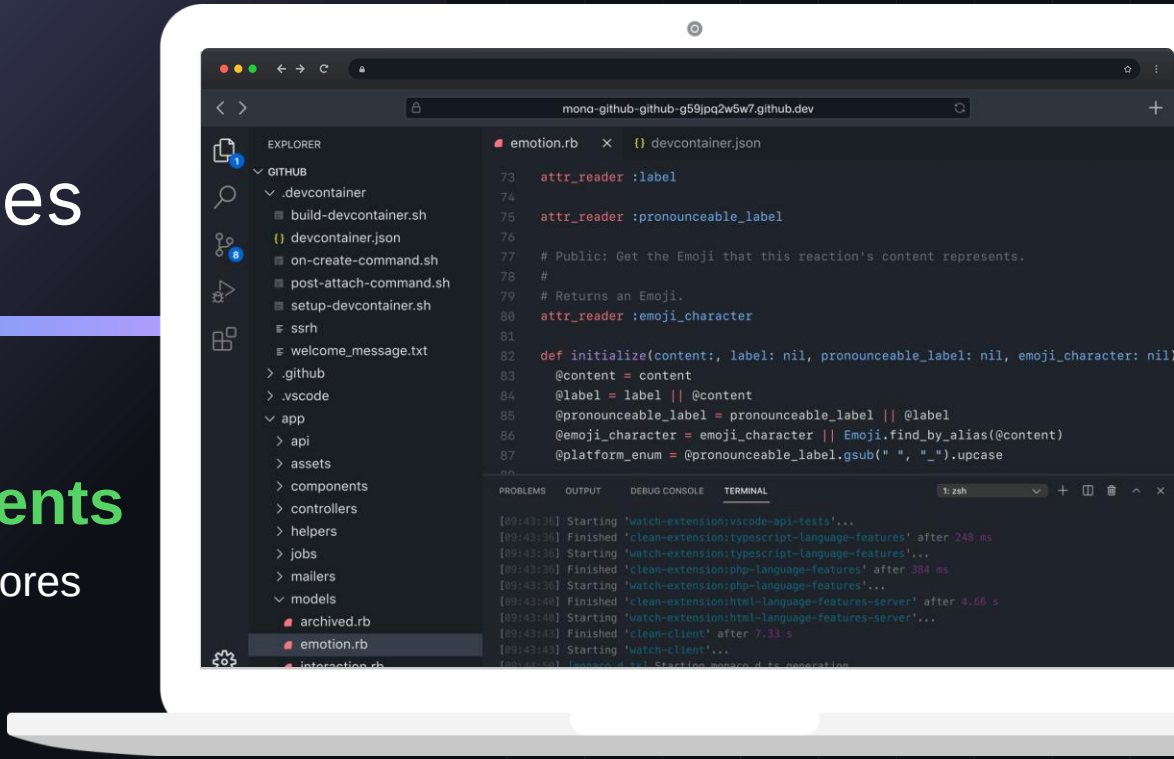
Teams continue to move workloads to the cloud and those that leverage all five capabilities of cloud see increases in software delivery and operational (SDO) performance, and in organizational performance.

Accelerate State of Devops Report 2021

GitHub Codespaces

Blazing fast cloud developer environments

Full compute backing, up to 32 cores and 64 GB RAM



Mean time to onboard

Faster bootstrap with a cloud-native experience

Improved auditing

Better visibility and traceability over local development

Standardized environments

Customize environments per repository, standardizing the development experience

Codespaces Lab

Create Node App and Test Coverage

- Create a new GH repo called **node-coverage-lab**
- Launch new Codespace on main branch(green button)
- When Codespace loads, run the following commands:
- **npm init –y**
- **npm install --save-dev jest**
- Verify node version: **node --version**
- Create new file **app.js**(add supplied code)
- Create a new folder at the root called **test**
- Create a new file **app.test.js(**add supplied code)
- Run git add ., git commit –m “ “, git push origin main
- Update package.json file to use Jest test case
- Run npm test

Update .devcontainer configuration

- At the root, create a new folder called **.devcontainer**
- Create a new file named devcontainer.json
- Add supplied configuration file
- Open command prompt(crtl+shift+p)
- Select “Codespaces: Rebuild Container”
- Verify Node version: node --version

GitHub Actions

GitHub Actions Fundamentals

Presented by GitHub Professional Services

Our Agenda

01
GitHub Actions
Introduction

02
Actions Workflow
Syntax

03
Actions Secrets
Environments and secrets

04
Manage Actions

05
Building Actions
Build Actions & Workflows

01
~~Migration~~
~~Migrate to Actions~~

02
Runners
GitHub Actions Runners

03
CI/CD
Action as CI/CD workflows

04
Labs & Demos
Actions Demos

GitHub Actions
Introduction

What's GitHub Actions

GitHub Actions is a core CI/CD & automation feature that operates natively within an integrated DevOps platform

- Workflows stored as yml files
- Fully integrated with GitHub
- Respond to GitHub events
- Live logs and visualized workflow execution
- Community-powered and custom workflows
- GitHub-hosted, self-hosted, or Kubernetes hosted runners
- Built-in secrets and environment variables store

Actions & SDLC

Use cases across SDLC using GitHub Actions

Plan
Tirage Issues and Project Boards using Actions

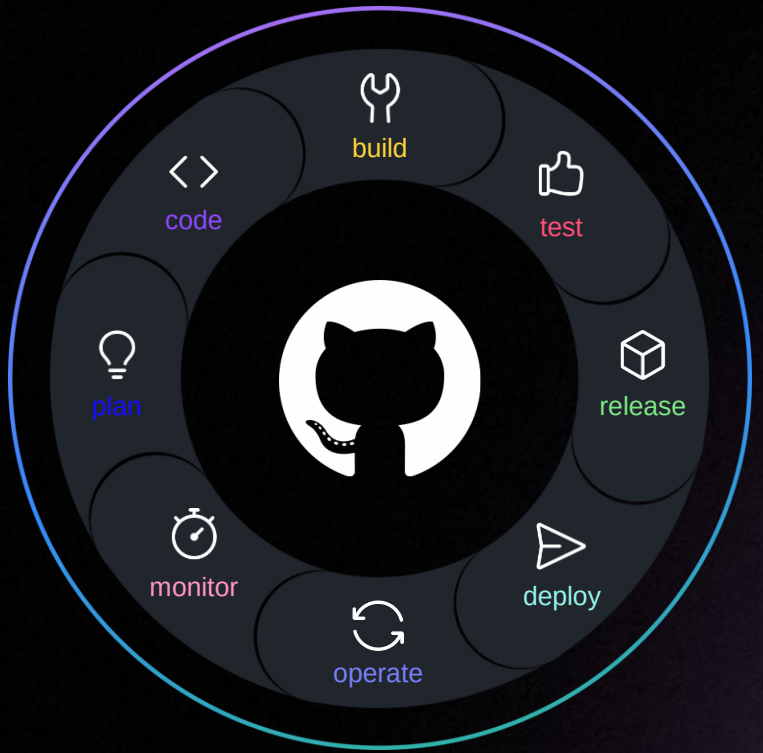
Code
Automates code builds, linting upon pull request creation

Build
Compile and build, Store artifacts

Test
Automotion of the unit tests, integration tests

Release
Publish release, Release notes, Versioning, and Upload artifacts

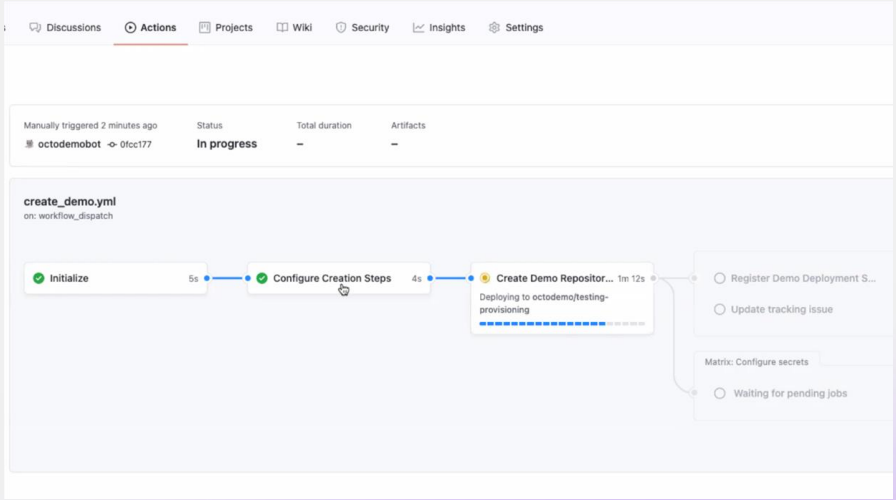
Deploy
Deploy applications to your chosen environment



Automation accessible to every developer

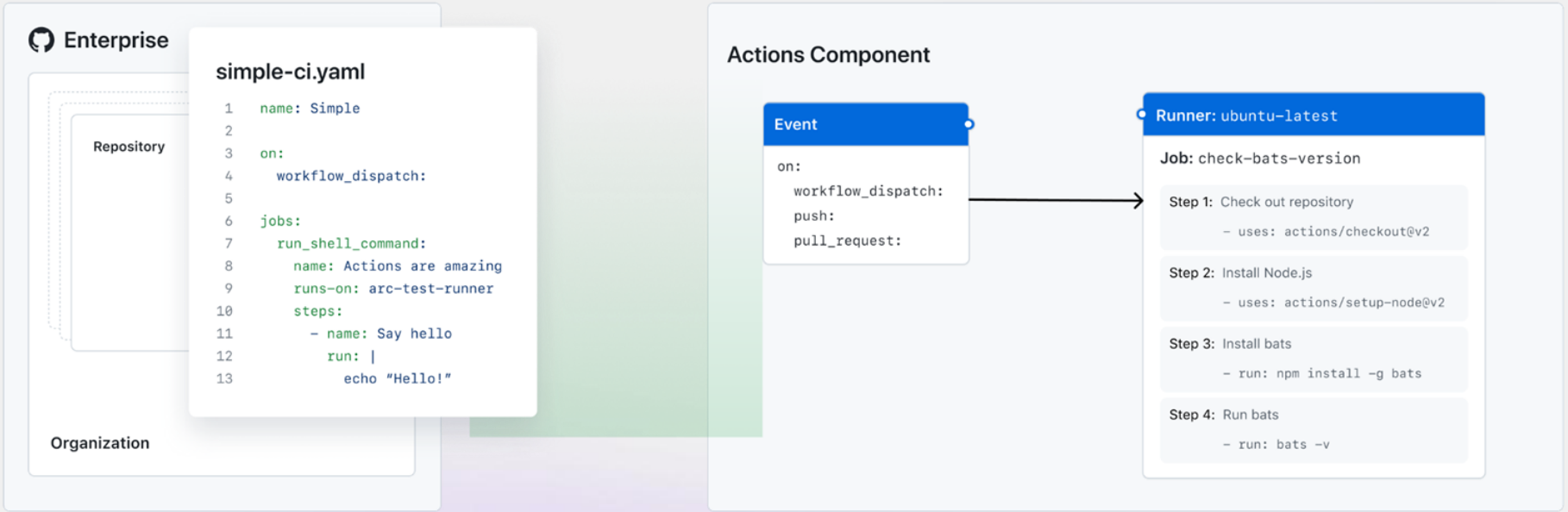
Actions

Enterprise-grade CI/CD that supports Windows, Linux, Mac



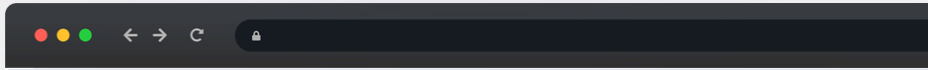
- Marketplace of 20,000+ Actions by our community
- Any operating system, cloud and on-prem
- Natively integrated into GitHub workflows
- Friction-free service or self-hosted runners
- Integrates Azure Pipelines & any other CI/CD
- DRY with Reusable Workflows
- API for viewing cache usage

GitHub Actions Key Components



Actions Workflow Syntax

Understanding the workflow file



The **name of the workflow** as it will appear in the "Actions" tab of the GitHub repository	`name: learn-github-actions`
The **name for workflow runs** generated from the workflow, which will appear in the list of workflow runs on your repository's "Actions" tab	`run-name: ${{ github.actor }} is learning GitHub Actions`
Specifies **the trigger for this workflow**. This example uses the push event, so a workflow run is triggered every time someone pushes a change to the repository or merges a pull request	`on: [push]`
Groups together all the **jobs** that run in the learn-github-actions workflow.	`jobs:`
Defines a job named **check-bats-version**. The child keys will define properties of the job.	`  check-bats-version:`
Configures the **job runner** on the latest version of an Ubuntu Linux runner.	`    runs-on: ubuntu-latest`
Groups together all the **steps** that run in the **check-bats-version** job and the uses keyword specifies that this step will run **v4** of the **actions/checkout** action	`    steps:` `      - uses: actions/checkout@v4`

Events

Webhook events

- Pull Request
- Push to repo
- Issues
- Releases
- ....

Scheduled events

Manual events

Events

Events

Webhook events

Scheduled events

Cronjobs

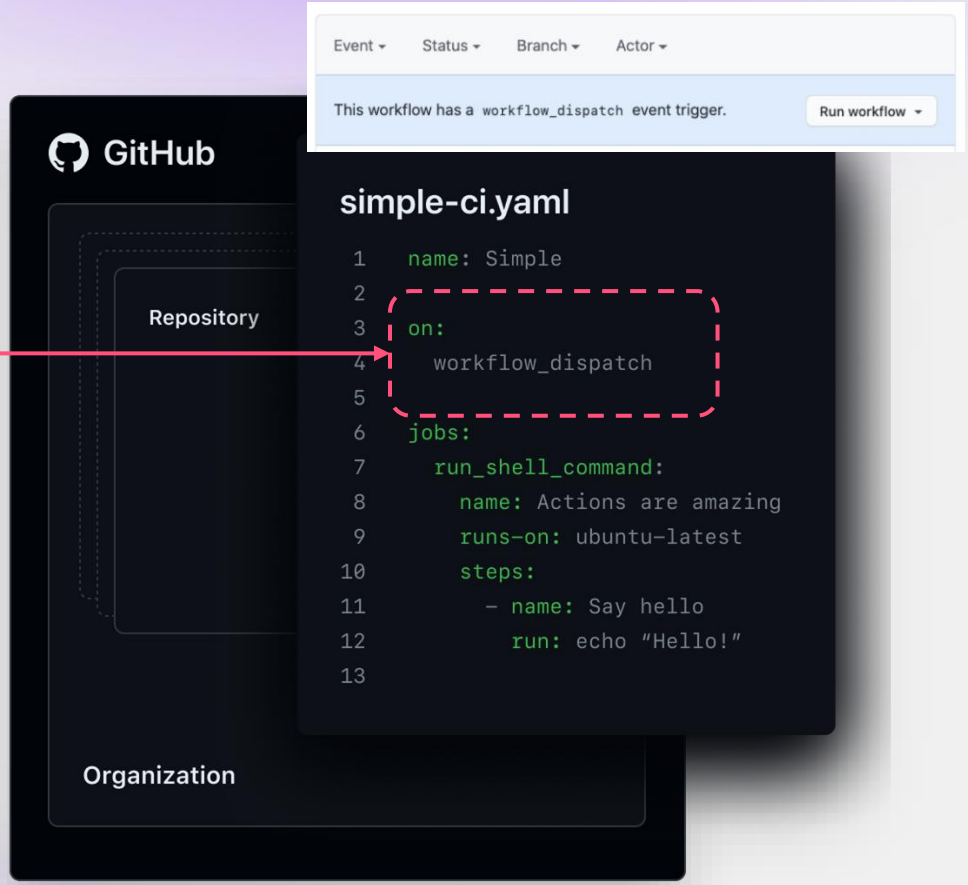
Manual events

Events

- **Webhook events**
- **Scheduled events**
- **Manual events**

Events

- workflow_dispatch
- repository_dispatch



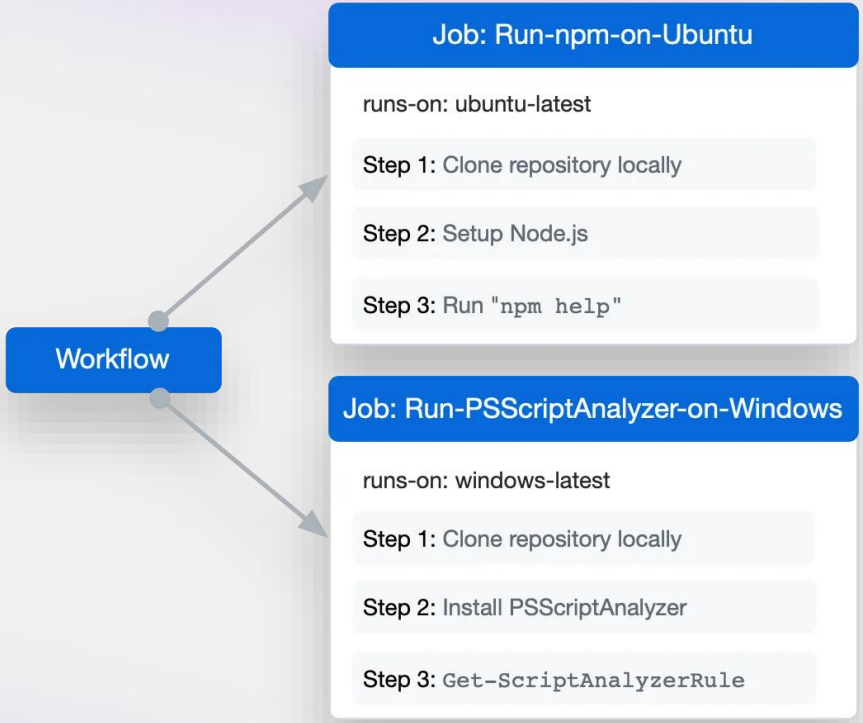
Runners

GitHub Hosted Runners

Self-Hosted Runners

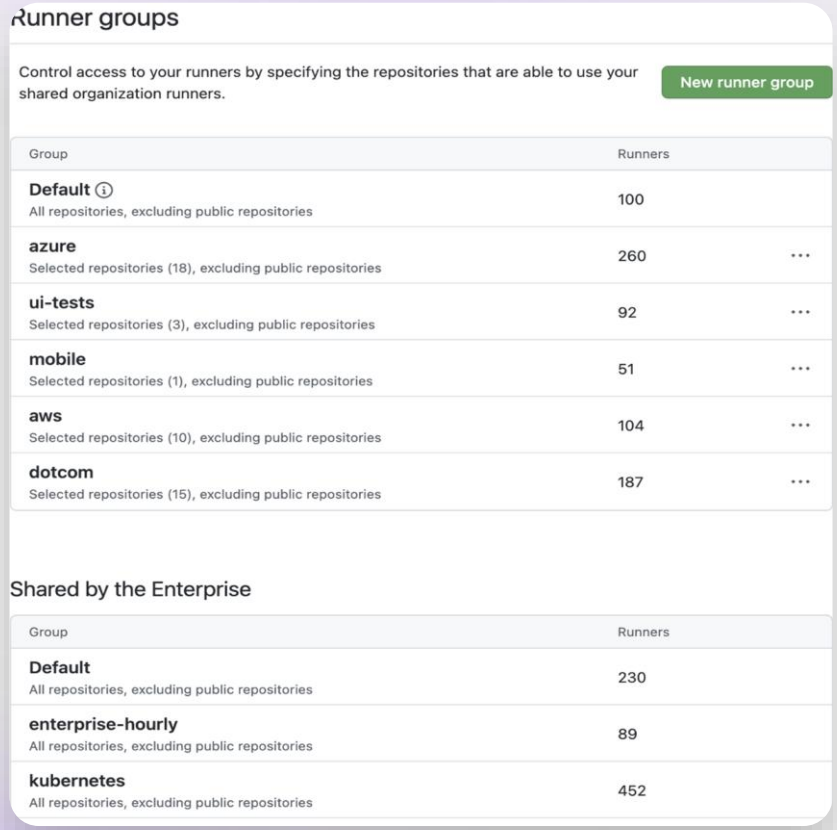
GitHub Hosted Runners

- Automatic Provisioning and Maintenance
- Diverse Operating System Support
- Integrated Software and Tools
- Scalability and Workflow Continuity
- Security and Administrative Privileges
- Customization and Local Environment Variables
- Using private vNet on Azure



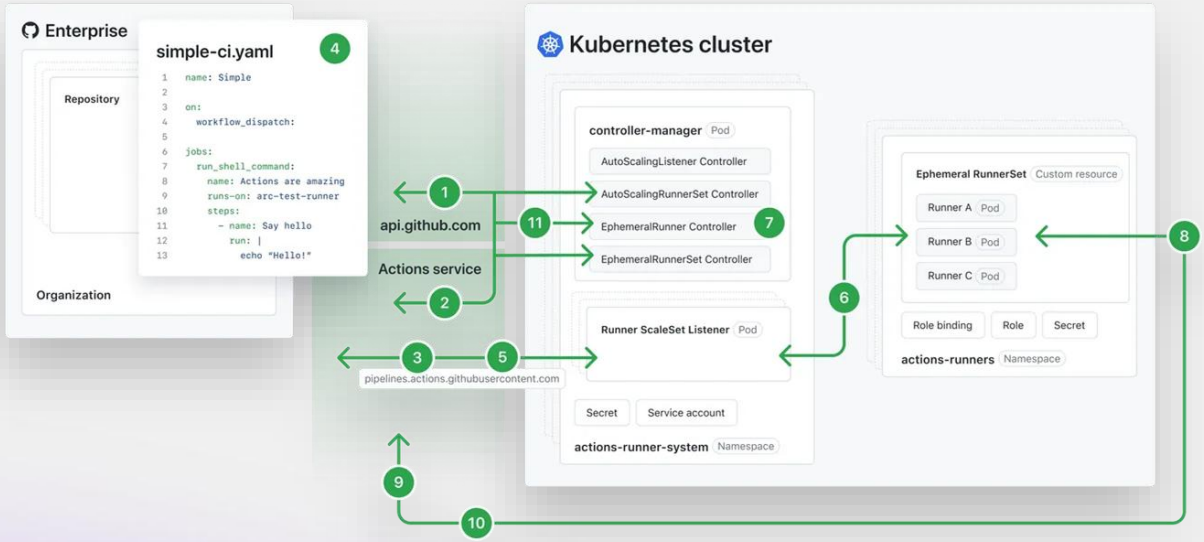
Self-Hosted Runners

- Full Control and Customization
- Flexibility in Hosting
- Cost Management
- Run on OS not supported on GitHub-hosted runner
- Usage Limits and Continuity
- Custom hardware config
- Can be grouped together



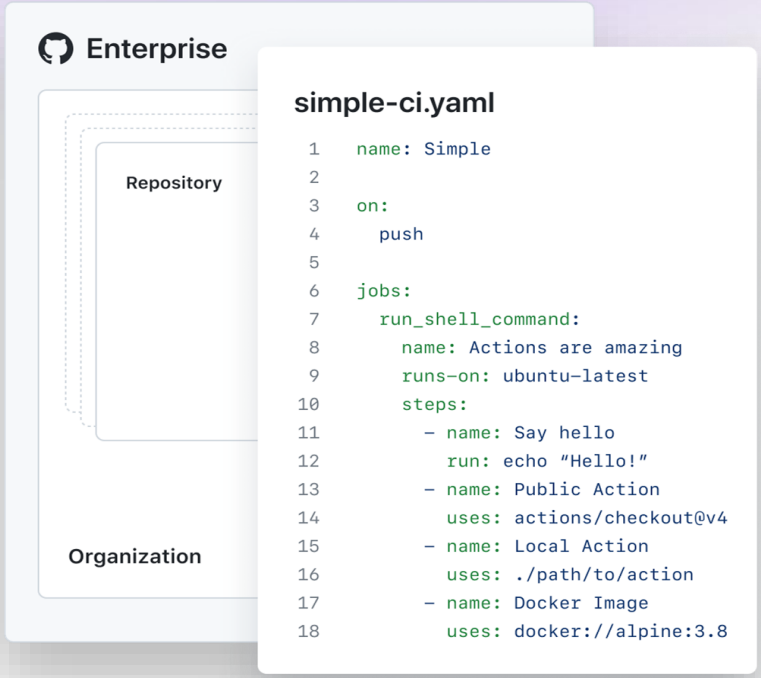
Self-Hosted Runners - ARC

- Kubernetes Integration
- Autoscaling Capabilities
- Ephemeral Runners
- Container-Based Runners
- Customizable Installation
- Runner Container Image



Actions

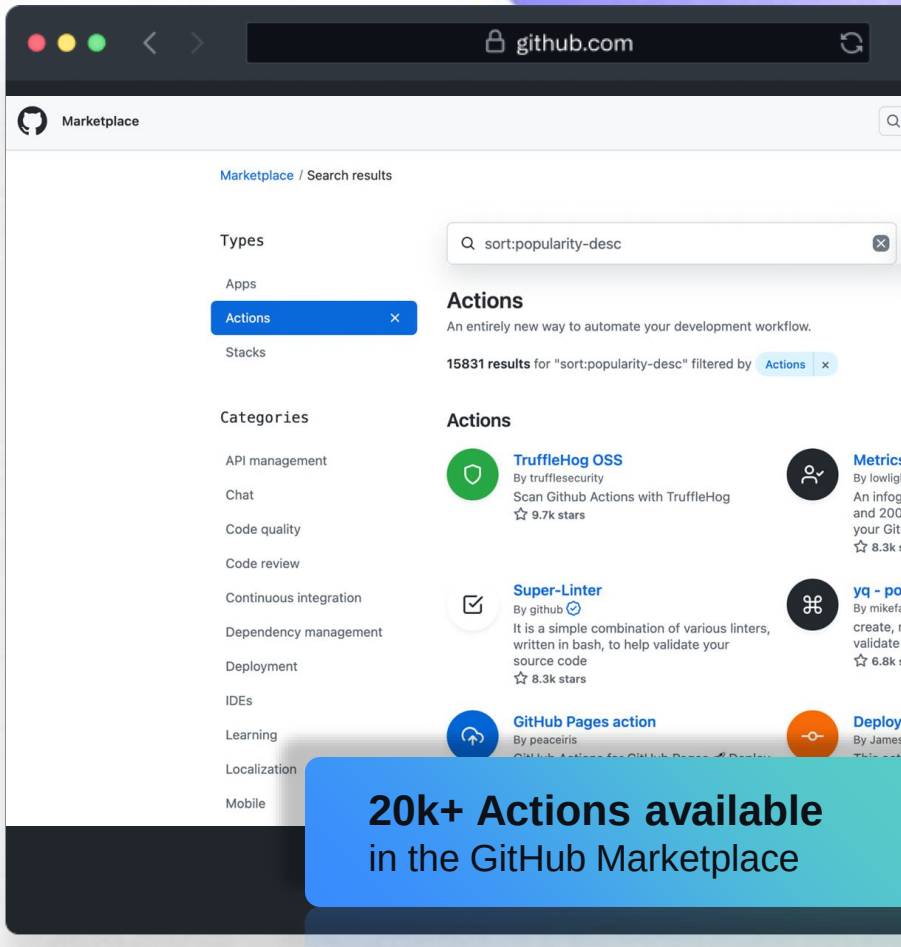
- Reusable units of code that can be referenced in a workflow
- GitHub runs them in Node.js runtime, or in containers
- Reference an Action, or run scripts directly



Script | Public Action | Local Action | Docker Image

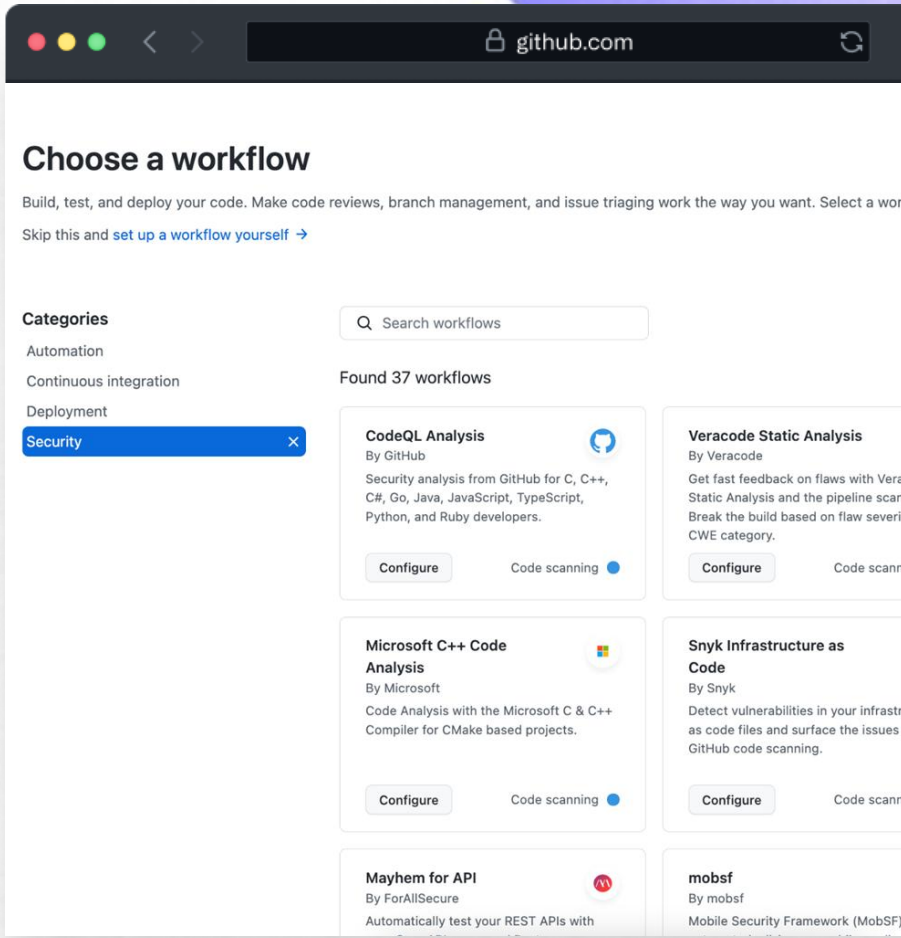
Largest Connected Developer Community

> “What I love about GitHub Actions is that you’re not limited in your choice of tools.

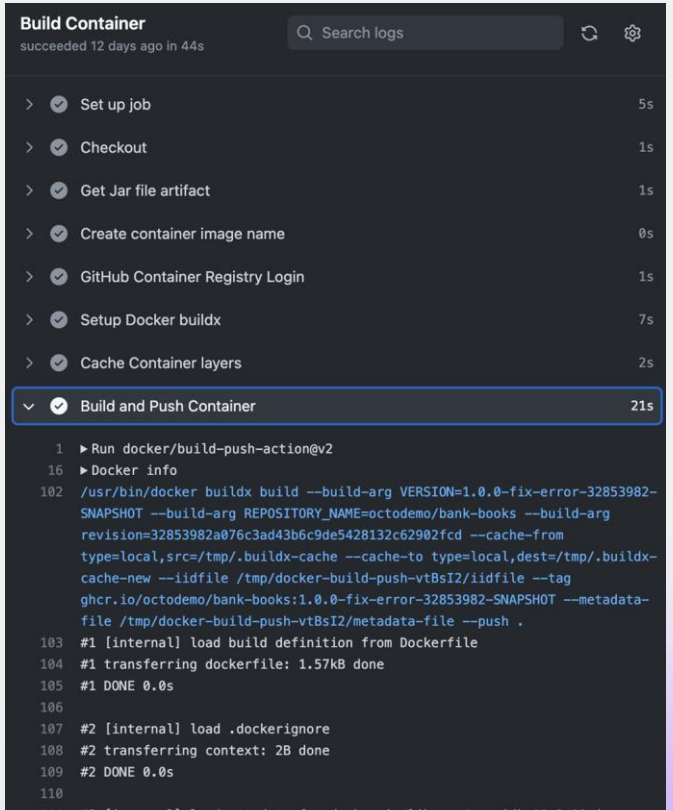


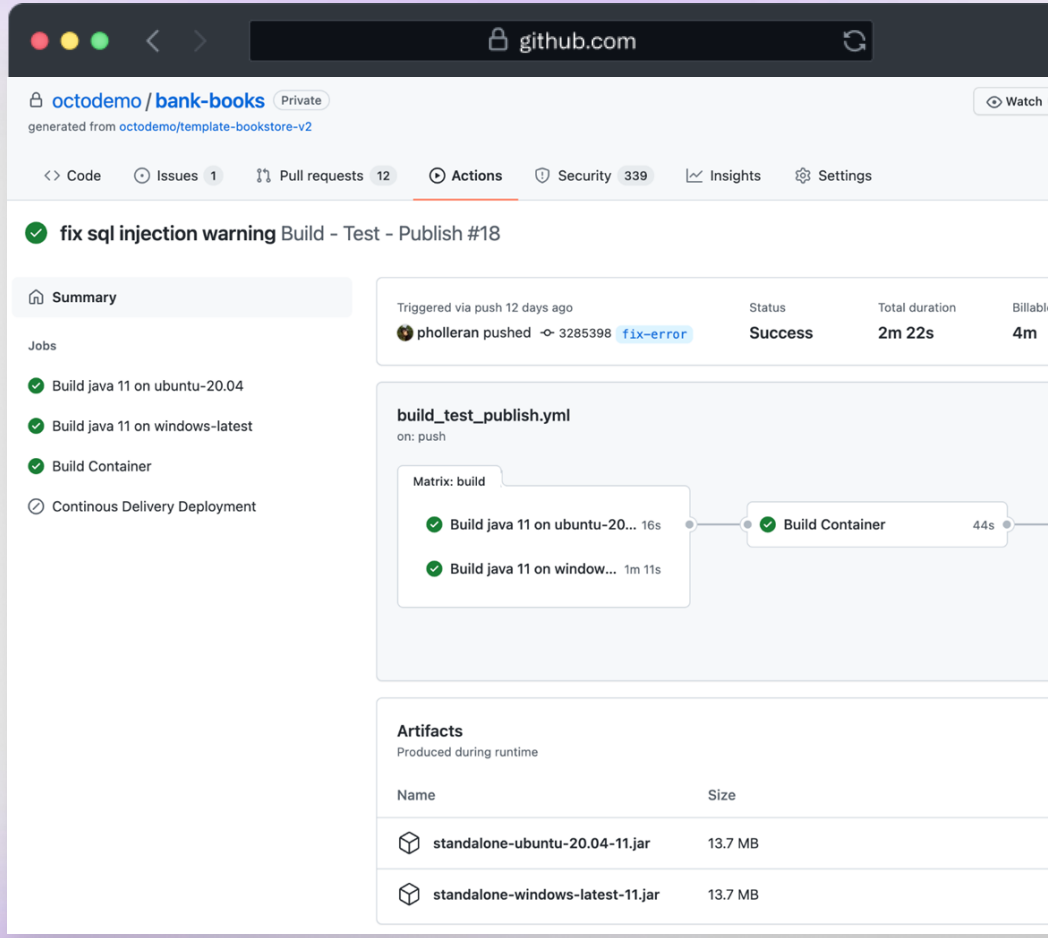
Starter workflows

- Preconfigured for specific languages and frameworks
- GitHub analyzes your code and suggests the workflows based on your language and framework
- You can also create your own starter workflow to share with your organization.



Workflow Logs





Advanced Syntax

permissions

Set workflow permissions for `GITHUB_TOKEN`

env

Set environment variables for all run steps

defaults

Set the shell and working directory for the run

concurrency

Manage workflows running concurrently

format

Format outputs

needs

Make jobs dependent of each other. Share outputs

if `success()` `always()` `cancelled()` `failure()`

Check whether a job should run based on variables.

timeout-minutes

Limit runtime

continue-on-error

Handle termination of workflows

join

Join arrays into strings

services

Create sidecar docker images for integration dependencies

container

Use a container for the steps execution

contains

Check if a string is contained in another

startsWith/endsWith

Check start/end of a string

toJSON/fromJSON

Make string JSON and JSON strings


Actions
Environments and secrets

Environments

Environments are used to describe a general deployment target like production, staging, or development.

Dynamic Creation

- Can be dynamically created based on the needs, allowing for flexible and scalable deployment targets.
- Automated setup and teardown of environments, ensuring efficient resource management

Review & Approval

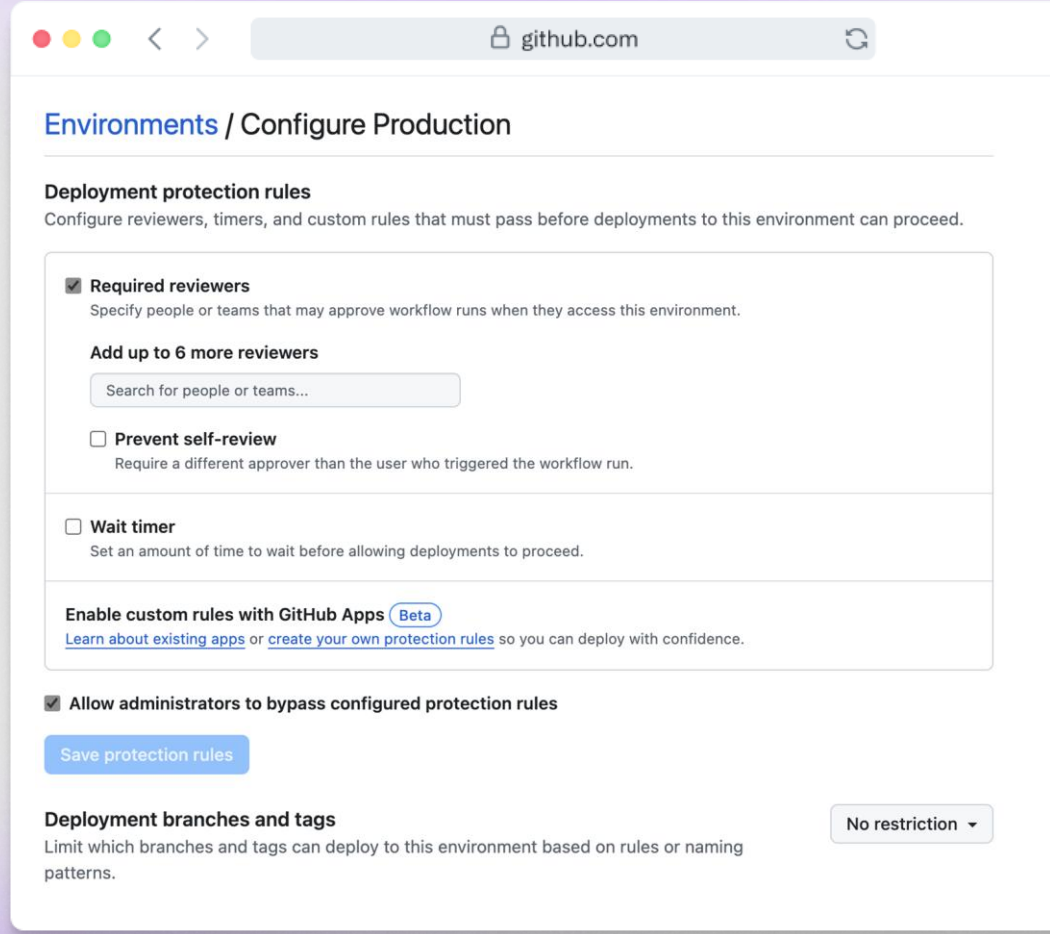
- Support a review and approval process
- Allows for designated reviewers to approve or reject deployments

Protection Rules

- Protection rules can be applied to environments, restricting access and enforcing policies
- Can include requirements such as specific branch protections, status checks

Environments manage and secure deployment targets like production

- Control deployments
- Add gated deployments with approvals
- Control secrets and envs variables
- Review all deployments
- Navigate directly to urls for deployments
- Fully integrated with the checks API
- Supports matrix for gated deployments



Secrets

GitHub Actions Secrets securely store sensitive data, manage access, and ensure encryption

Storage & Access

- Allow you to store sensitive information securely at the repo, environment, and org levels
- Access to secrets can be controlled through policies

Encryption & Security

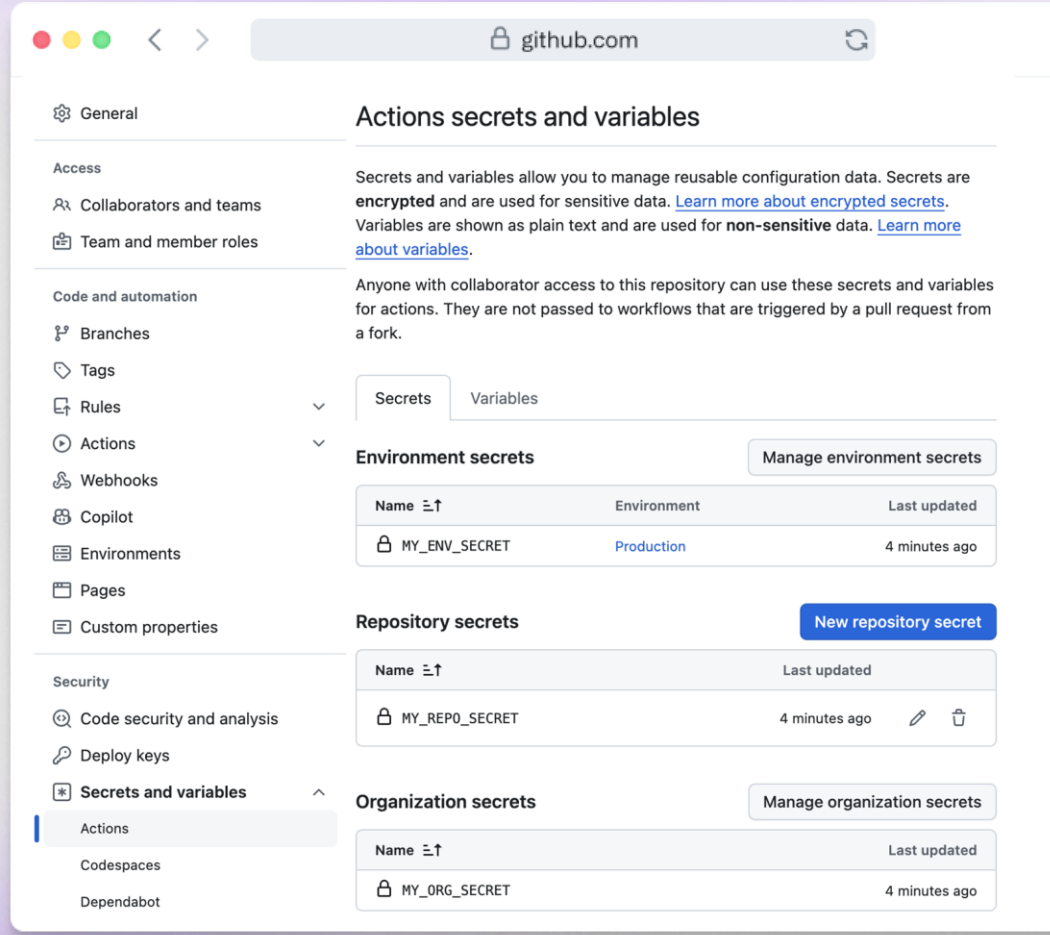
- Secrets are encrypted before storage & during transmission and storage
- Redacts secrets from logs and allows to use short-lived tokens

Integration & Management

- Secrets can be integrated into workflows by setting them as inputs or environment variables
- Management of secrets is facilitated through the GitHub REST API and GitHub CLI

secrets ensure secure handling of credentials and sensitive data across various scenarios

- Built-in secret store
- Encrypted (LibSodium sealed box)
- Use directly from your workflow
- Redacted in workflow logs
- API & CLI support
- Organization / repository / environment secrets



Environment secrets

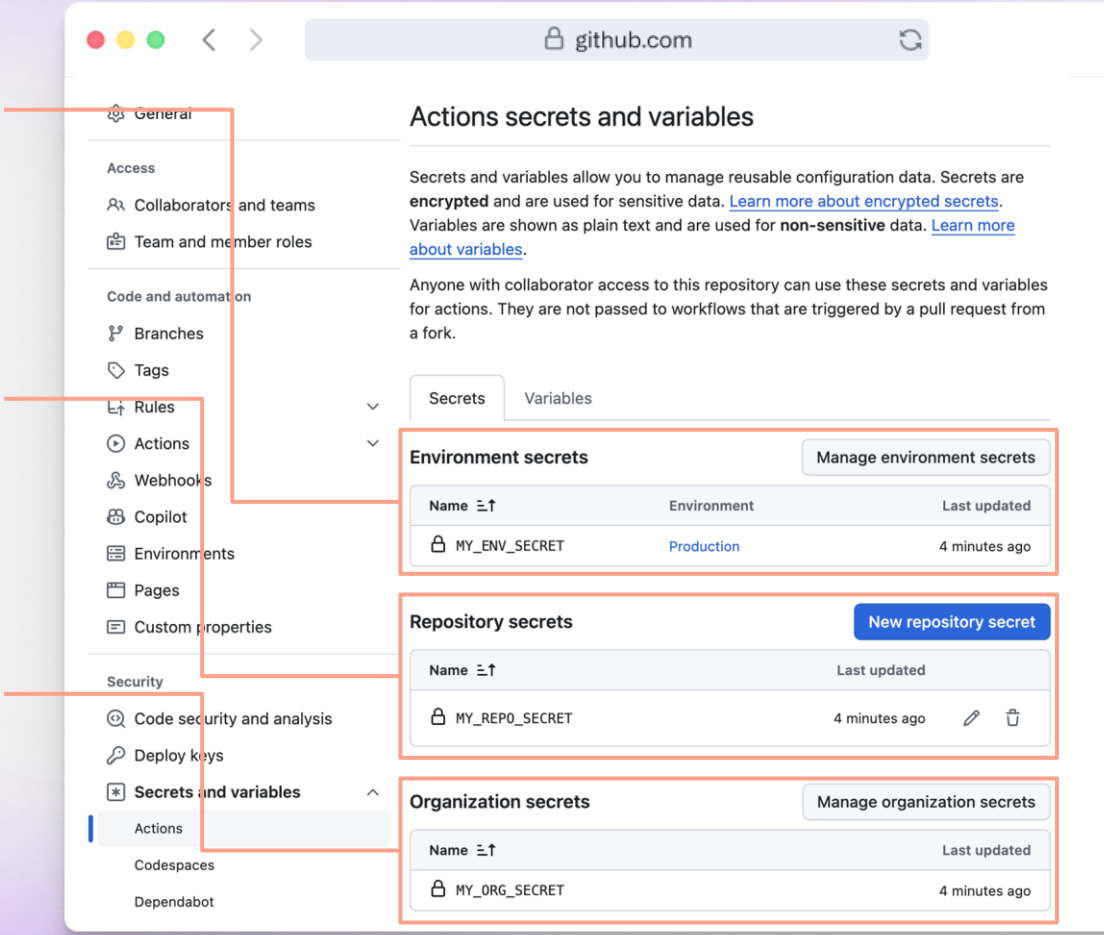
- Built-in secret store
- Encrypted (LibSodium sealed box)

Repository secrets

- Use directly from your workflow
- Redacted in workflow logs

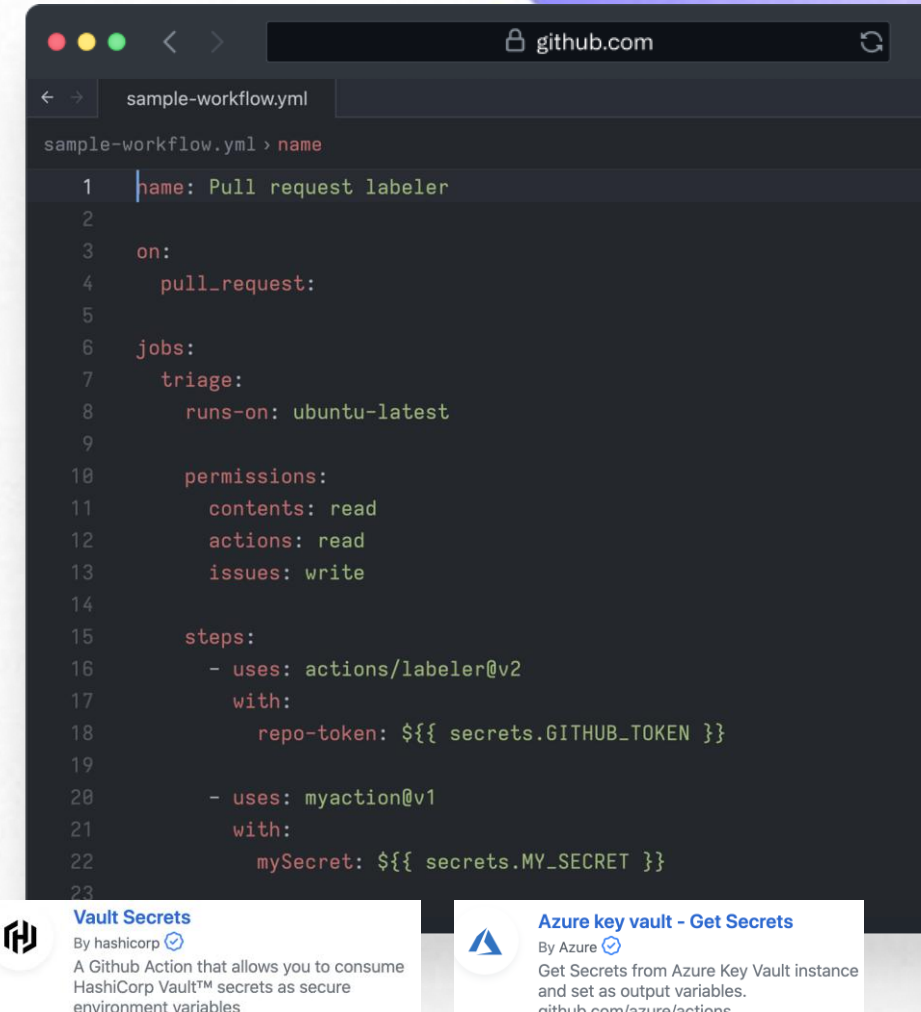
Organization secrets

- API & CLI support
- Organization / repository / environment secrets



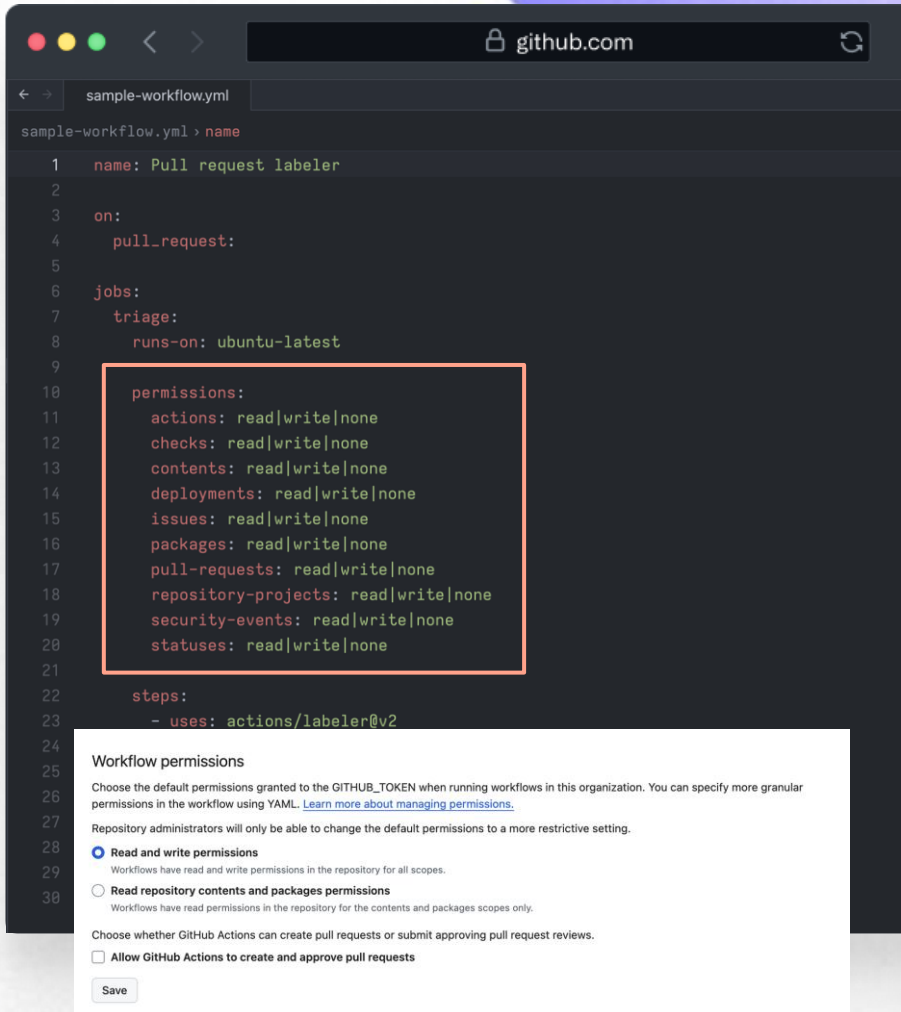
Using secrets in workflows

- All secrets can be accessed using the same syntax;
  - `${{ secrets.<SECRET_NAME> }}`
- Every workflow run provisions a `GITHUB_TOKEN` secret by default
  - Scoped to a single repository
  - Enterprise/organization/repository policies for default permissions
  - **`permissions`** syntax for granular permissions on workflow- or job-level
  - Can't trigger other workflows
- Marketplace Actions exist for integration with other secret stores



Permissions for GITHUB_TOKEN

- **Token Creation**: Automatically created for each workflow job
- **Token Scope**: Limited to the repository containing the workflow
- **Modify Permission**: Adjustable in workflow files for specific needs
- **Default Permissions**: Admins can set to permissive or restricted
- **Additional Permissions**: Use GitHub App or personal access tokens




Building Actions
Build Actions &
Workflows

Write Custom Action

- Actions Types
  - JavaScript Actions
  - Docker Actions
  - Composite Actions
- Repository Structure
  - action.yml
  - Dockerfile
  - Source code files

```yaml
name: "Hello Action"
description: "Greet someone"
author: "octocat@github.com"

inputs:
 my_name:
   description: "Who to greet"
   required: true
   default: "World"

outputs:
 greeting:
   description: "Full greeting"

runs:
 using: "docker"
 image: "Dockerfile"

branding:
 icon: "mic"
 color: "purple"
```

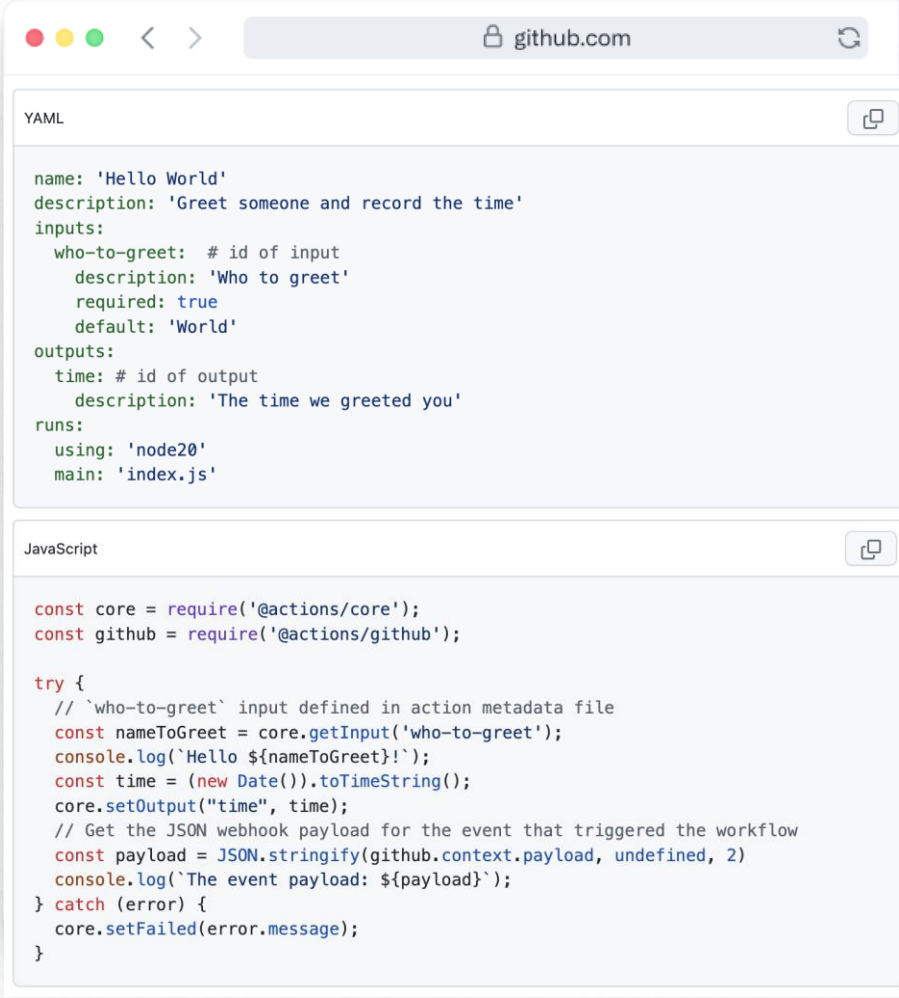
JavaScript Action

A JavaScript action is a custom action written in JavaScript or TypeScript that runs directly on the GitHub-hosted runner or a self-hosted runner. It allows you to leverage the full power of Node.js along with GitHub's rich API and automation capabilities. JavaScript actions are ideal for tasks that require high performance and need to interact directly with the GitHub environment.

action.yml Metadata File

index.js Action Code

package.json Package Configuration



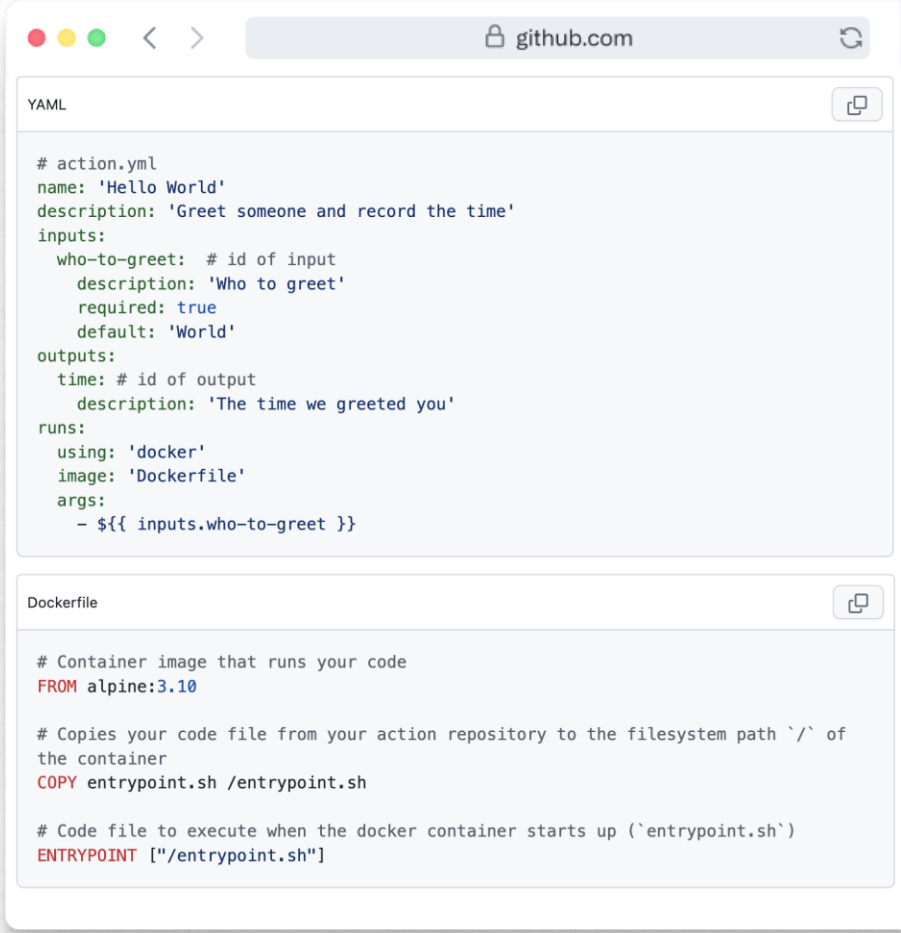
Docker Action

A Docker Action is a custom GitHub Action that runs inside a Docker container. This type of action leverages the capabilities of Docker to create a portable and reproducible environment. Docker Actions are ideal for scenarios where you need a specific software environment, have complex dependencies, or require a high level of isolation.

Dockerfile — Dockerfile for environments

action.yml — Action Metadata File

entrypoint.js — Entrypoint Script

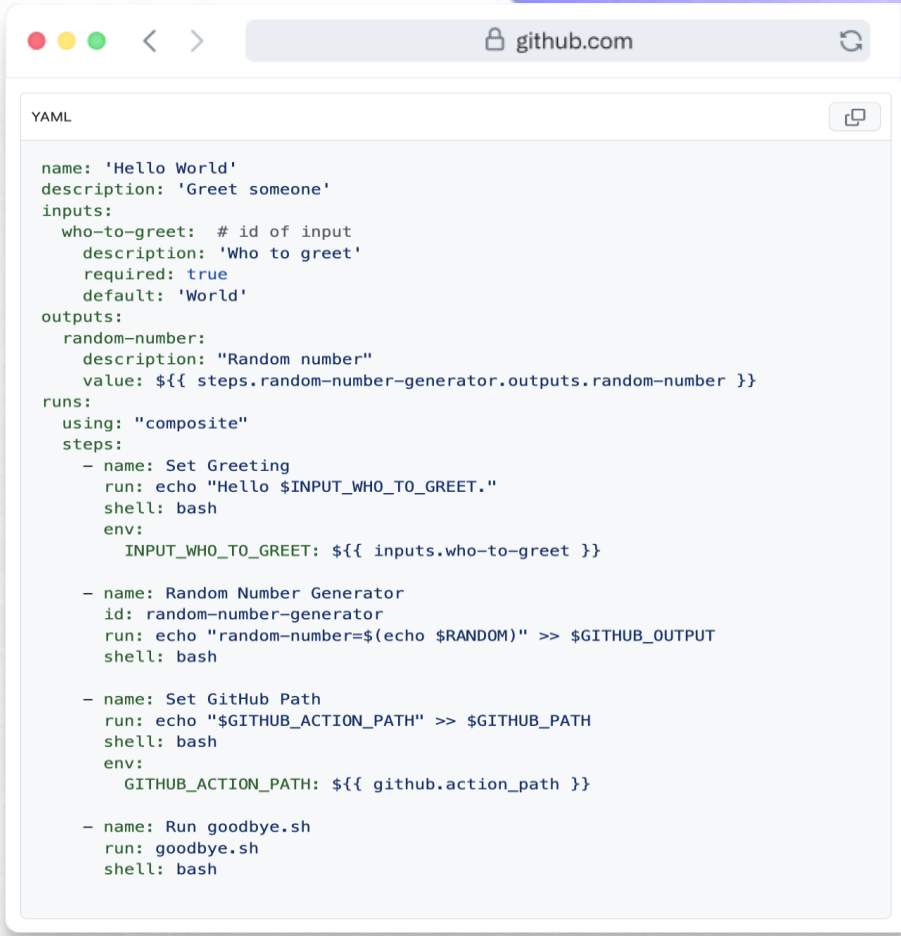


Composite Action

A Composite Action in GitHub is an action that combines multiple individual steps into one reusable component. Instead, they use YAML syntax to define a sequence of steps that are executed together as a single unit. This allows you to encapsulate common tasks and reuse them across different workflows, improving maintainability and reducing redundancy.

action.yml — Metadata File

Steps — A series of steps defined in the `action.yml`



Best Practices

Write your own Action

Creating your own GitHub Actions can significantly enhance the automation of your CI/CD pipelines and improve your workflow efficiency.

Security

- Limit Permissions
- Pin Dependencies
- Handle Secrets Securely

Documentation

- README File
- Provide Examples

Testing

- Local Testing
- Validation

Reusability

- Reusable Components
- Composite Actions
- Use GitHub Actions Toolkit

Optimization

- Efficient Code
- Caching

Runners

GitHub Actions Runners

GitHub Runners

GitHub Hosted Runners

- **Managed Environment**
  - Maintenance
  - Pre-Installed Software
- **Resource Allocation**
  - Scalability
  - Cost
- **Security**
  - Isolation
  - Network Access
- **Ease of Use**
  - Quick Setup
  - Limited Customization
- Performance
  - Standard Performance

Self-hosted Runners

- **Custom Environment**
  - Full Control
  - Customization
- **Resource Allocation**
  - Scalability
  - Cost
- Security
  - Responsibility
  - Network Access:
- Ease of Use
  - Setup Complexity
  - Flexibility
- Performance
  - Custom Performance

Self-Hosted Runners

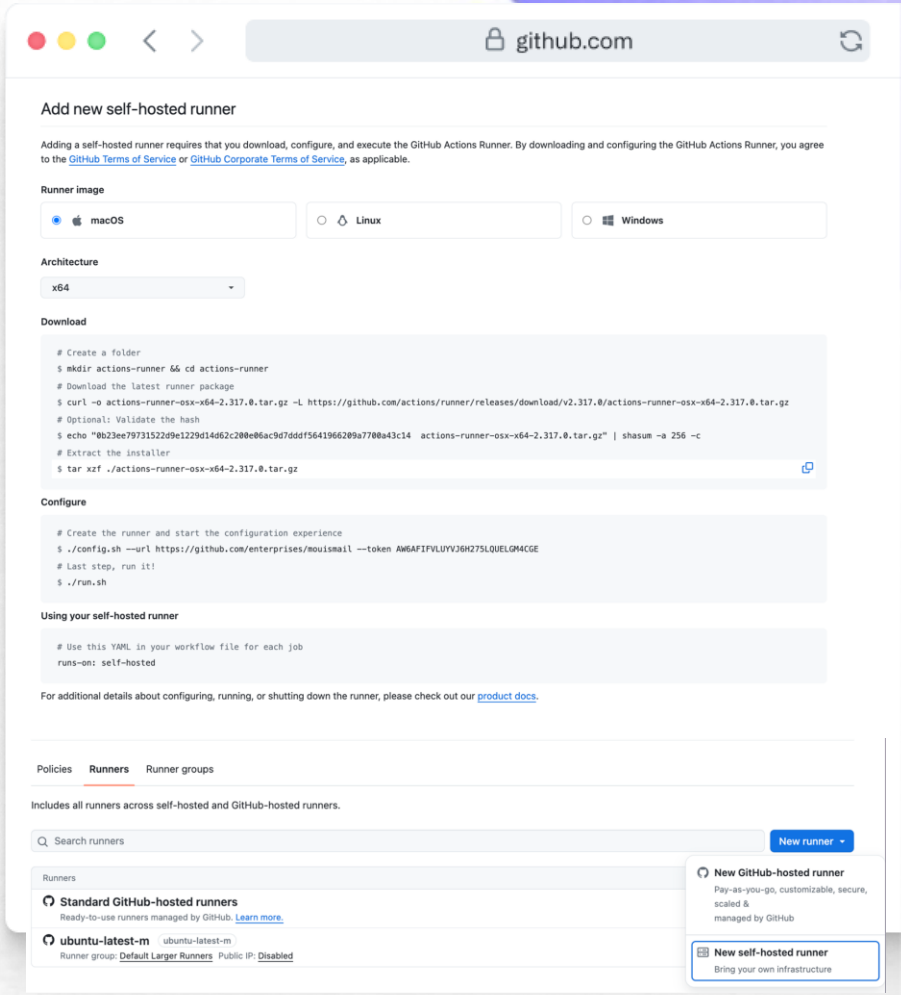
Self-hosted runners are machines that you manage and maintain to execute jobs from GitHub Actions workflows.

Provision the Runner

Install the Runner Application

Configure the Runner

Manage and Maintain



Security
With Self-hosted Runners

Using self-hosted runners for GitHub Actions offers greater flexibility and control over the environment, but it also introduces additional security considerations.

Runner Isolation

- Isolation of Runners
- Ephemeral Runners

Network Security

- Restricted Network Access
- VPN and VPC

Access Control

- Least Privilege Principle
- Access Tokens

Environment

- Secure Configuration
- Monitoring and Logging

Data

- Secrets Management
- Encryption

Best Practices
With Self-hosted Runners

> **Securing self-hosted runners for GitHub Actions requires careful planning and adherence to best practices**

Runner Lifecycle Management

- Automate Provisioning
- Automate Updates

Containerization

- Use Containers
- Custom Container Images

Ephemeral Runners

- Short-Lived Runners
- Scale on Demand

Audits

- Conduct Audits
- Compliance Checks

CI/CD

Action as CI/CD workflows

Basic CI/CD Action

GitHub Actions is a powerful feature within GitHub that enables the automation of workflows, particularly for continuous integration (CI) and continuous deployment (CD)

Build & Test

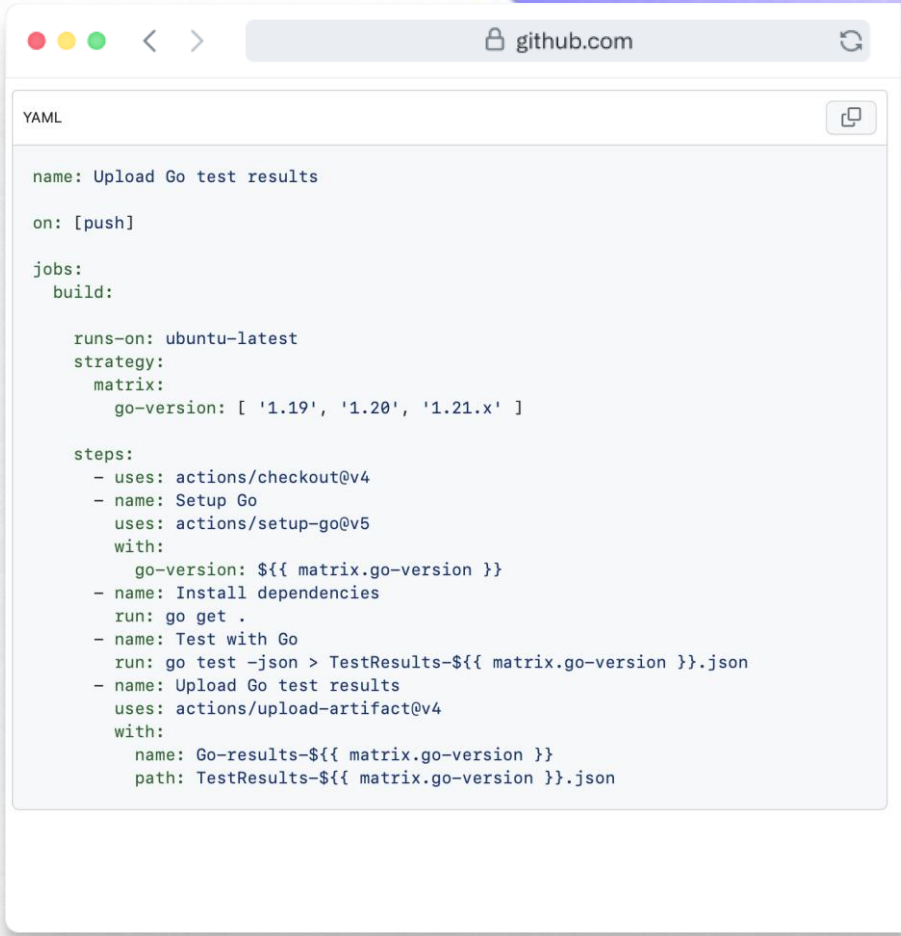
This guide shows you how to build, test, and publish a Go package

Specifying a Go version

Installing dependencies

Caching dependencies

Building and testing your code



Deploy

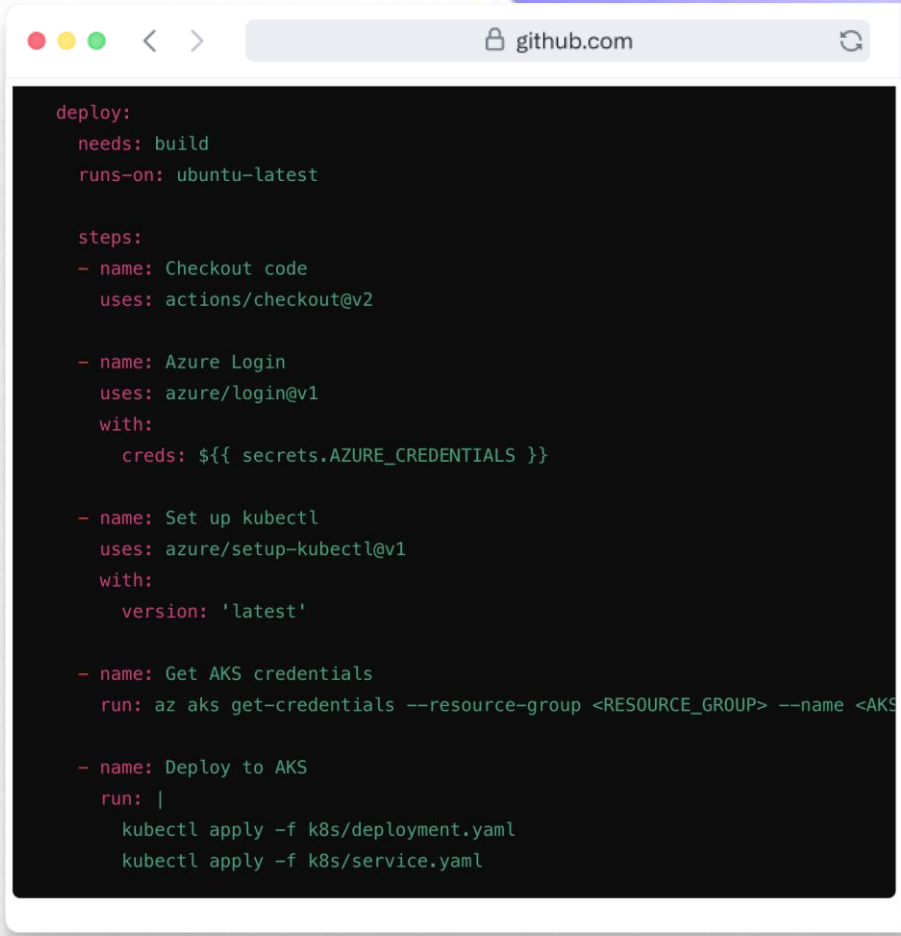
This guide explains how to use GitHub Actions to build and deploy a project to Azure Kubernetes Service

Azure Login

Build image on ACR

Configure deployment

Deploys application



GitHub Actions Lab

GitHub Actions Lab

- Navigate to **node-coverage-lab** repo
- Select “Actions” tab
- Seach for node project and select “Node.js” and click configure
- Add step to upload coverage test artifacts(for node version metrix) to repo
- Edit GH Actions file(node.js.yml) directly in browser and commit changes
- Ensure job completed successfully

Create Node App and Test Coverage

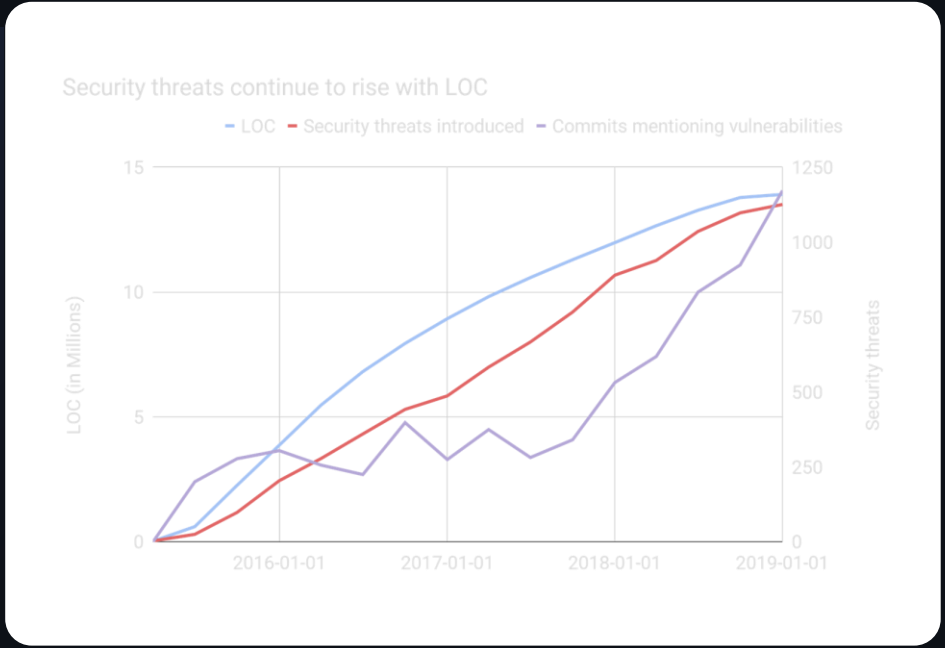
- Create a new GH repo called **node-coverage-lab**
- Launch new Codespace on main branch(green button)
- When Codespace loads, run the following commands:
- npm init –y
- npm install --save-dev jest
- Verify node version: node --version
- Create new file **app.js**(add supplied code)
- Create a new folder at the root called **test**
- Create a new file **app.test.js(**add supplied code)
- Run git add ., git commit –m “ “, git push origin main
- Run npm test

GitHub Advanced Security

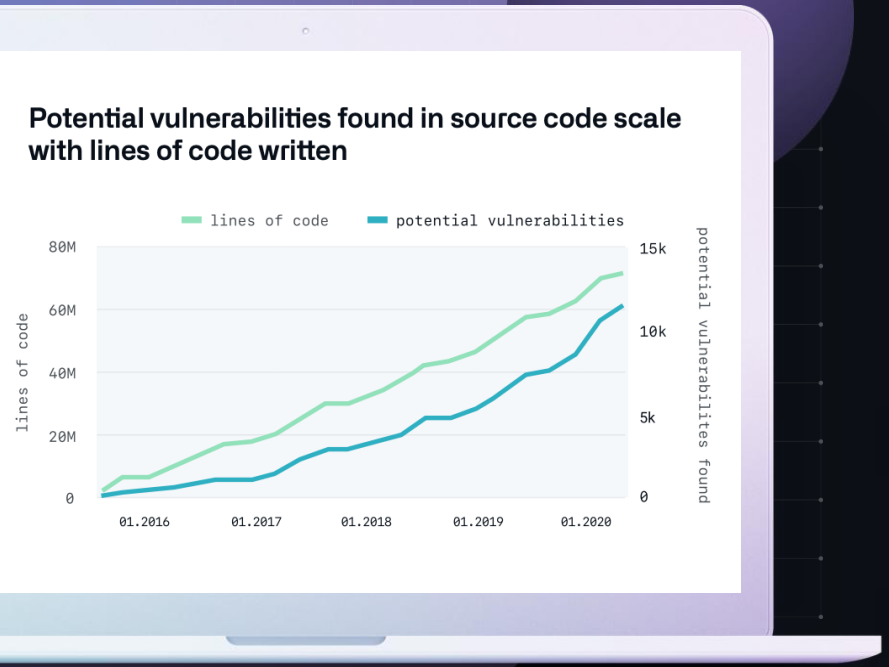
Challenges and Opportunities

Industry challenges and insights

Despite increasing developer awareness, security threats continue to rise



State of AppSec



Despite billions of dollars of investment…

- 85% of applications still contain a security issue
- Code written in 2020 is just as likely to introduce a security issue as code written in 2016

State of AppSec

Flaws in applications are consistently the #1 attack vector for breaches

is falling further behind the current state of Development

1:100 Security team members to developers

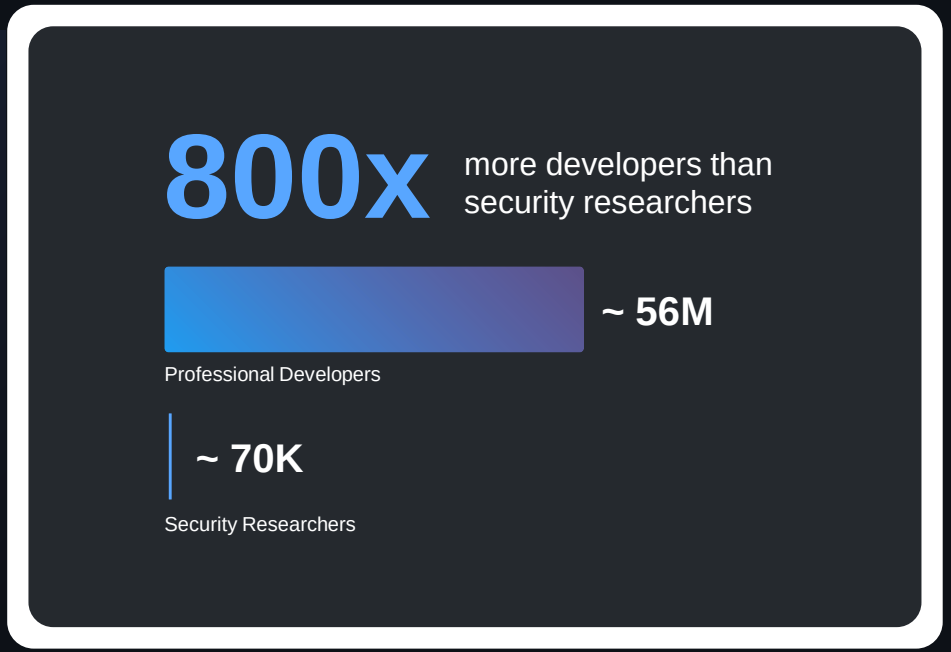
Lack of knowledge voted the main AppSec challenge

Remediation trends are stagnant

Everyone wants to shift security left…

Solution:

Shift security left, but how?



The GitHub Difference

Developer empowered security to fix in minutes not months

Native

Embedded from init to ship

Automatically deployed

Detailed changelog

Visibility into security posture across all code

Trust-By-Design

Static code analysis

Supply chain analysis

Secret scanning

Security Lab

Extensible platform

Fix in minutes

In-context feedback

Automated fixes

Jargon free language

Blameless culture

Community Driven

Supported by millions of developers and security researchers

Contributions from leading peers

Native Application Security

Secure code without disrupting innovation with security testing embedded in the developer experience

> Security at the expense of usability comes at the expense of security.
>
> **Lee Cookson**
> Engineering Director |
> @ Dow Jones

- **Embedded from init to ship**
- **Automatically deployed** to immediately increase code coverage
- **Detailed changelog** demystifying progress and simplifying audit/collaboration
- **Security overview** provides visibility into security posture across all code

Trust-by-design

> GitHub helps us ensure that we have our security controls baked into our pipelines all the way from the first line of code you're writing.
>
> **Miguel El Lakkis**
> Chief Information Security Officer | @ Dow Jones

Trust your software with an end-to-end security solution for securing your software supply chain and code, that developers actually use and love!

Static code and supply chain analysis

Secret detection and leak prevention

Extensible platform with SARIF support

Fix in Minutes

Our engineering teams place tremendous value in GitHub’s latest security features like vulnerability alerts and automated fixes.

Jon Parise
Engineering Architect |
@ Pinterest

Empower developers with more accurate results and fix the vulnerabilities that matter the most for your organization

In-context feedback and automated fixes where possible leading to industry leading fix rates and MTTR

Jargon free language that empowers developers to fix without security intervention

Blameless developer trusted culture that leads to higher adoption of fixes

Community Driven

Benefit from thousands of analyses created by GitHub's team of security researchers and language experts and contributions from other GitHub users- such as leading security researchers at Microsoft, Google, Uber, and others.

Scale your security team with the World's Leading Security Researchers

Scale your security team with a **dedicated security team that also leverages the world's expert community of developers**

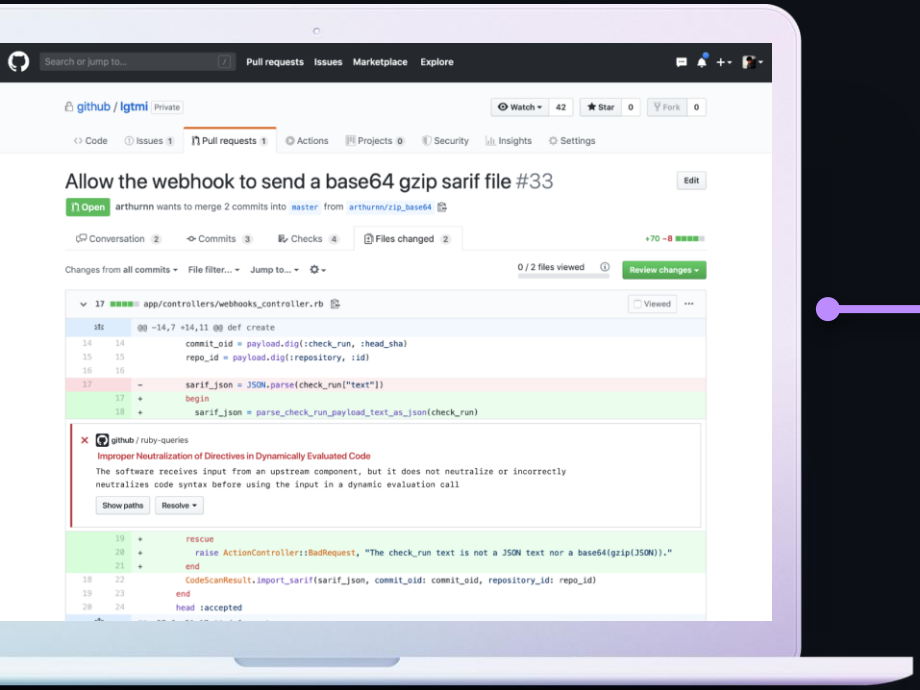
Open source foundations **encourage collaboration and community engagement**

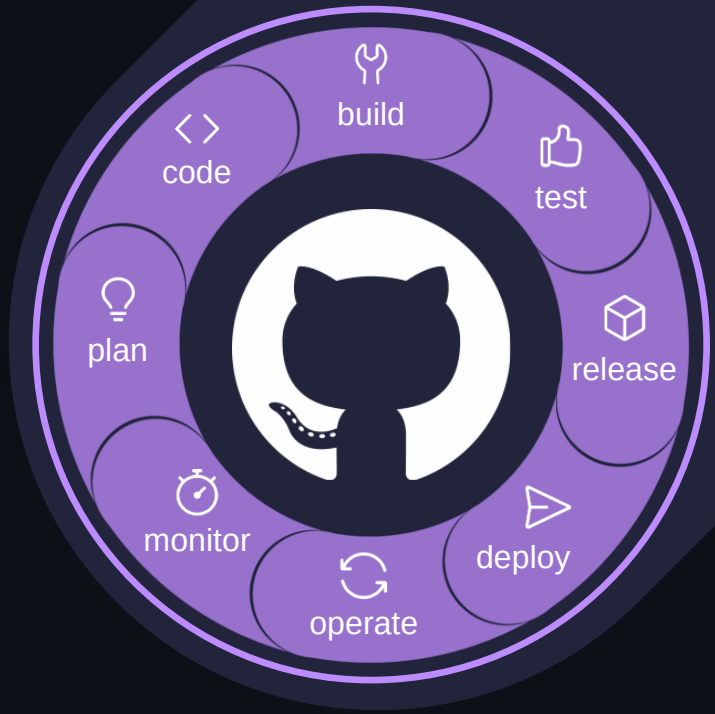
Benefit from user driven design and frequent innovation

Secure by design

Secure development end to end

Security built into the Developer Lifecycle





Community-powered security and compliance

Dependency insights

Real-time inventory

License compliance

Vulnerability alerting

Vulnerability Management

Automated code scanning

Private secret scanning

Largest vulnerability database

Automated security updates

CodeQL

World’s most advanced code analysis

Vulnerability hunting tool

Community of top security experts

GitHub Advanced Security Impact

Code vulnerability fix rate of

72%

compared to

15%

Industry norm after 7 days

OSS vulnerabilities MTTR decreased from 180 days to 40 days with Dependabot

GHAS scan and alerts for secrets and tokens from 60+ providers and growing

Dependabot

Automated pull requests for security & version updates

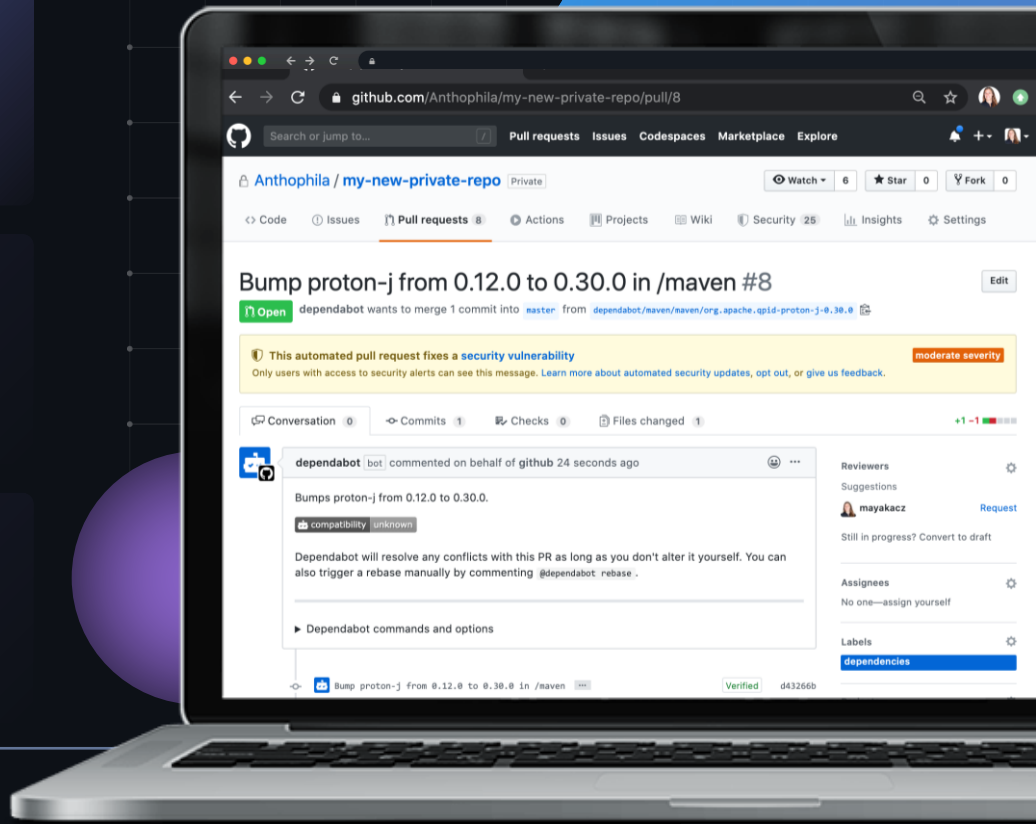
Keep your projects secure and up to date by monitoring them for vulnerable and out-of-date components. If a suggested update is found, we'll automatically open a pull request with suggested fixes.

Integrated with developer workflow

Dependabot is integrated directly into the developer workflow for a frictionless experience and faster fixes.

Rich vulnerability data

GitHub tracks vulnerabilities in packages from supported package managers using data from security researchers, maintainers, and the National Vulnerability Database—all discoverable in the GitHub Advisory Database.



Code scanning

Find and fix vulnerabilities fast

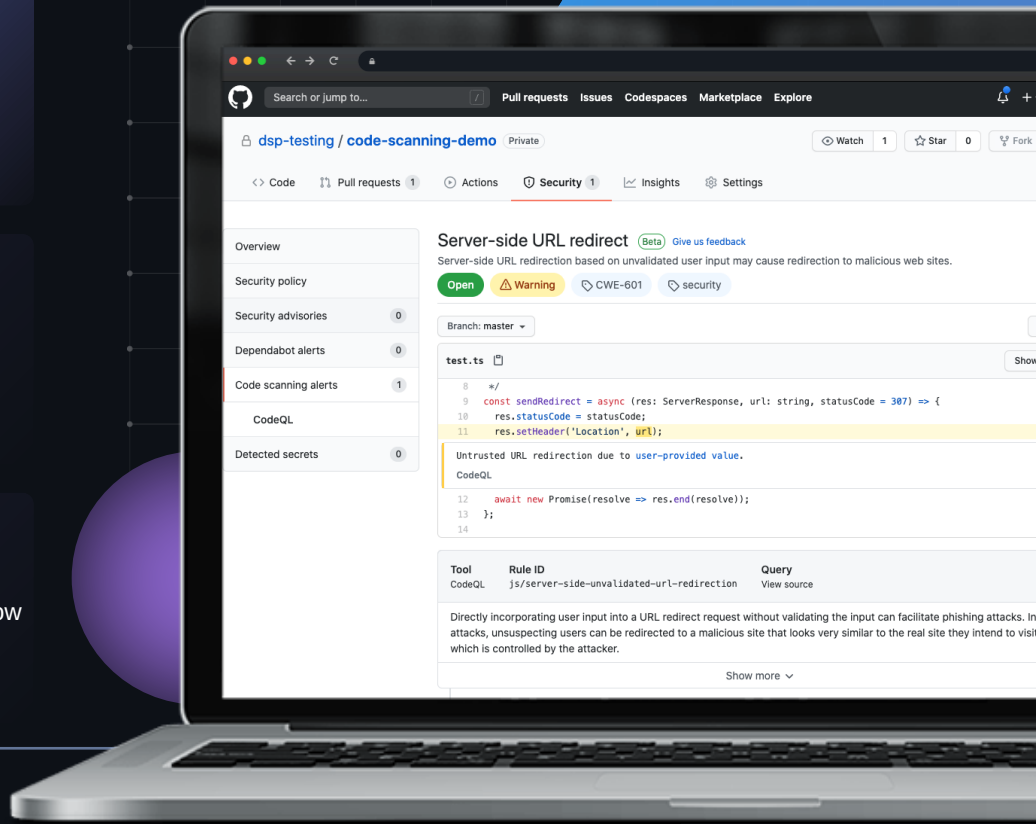
Find and fix vulnerabilities before they are merged into the code base with automated CodeQL scans.

Community of top security experts

Your projects are powered by world-class security teams. Use queries created by the security community in your projects.

Integrated with developer workflow

Integrate security results directly into the developer workflow for a frictionless experience and faster development



Secret scanning

Identifies secrets as early as possible

Finds secrets (including Azure secrets) the moment they are pushed to GitHub and immediately notifies developers when they are found.

Community of secret scanning partners

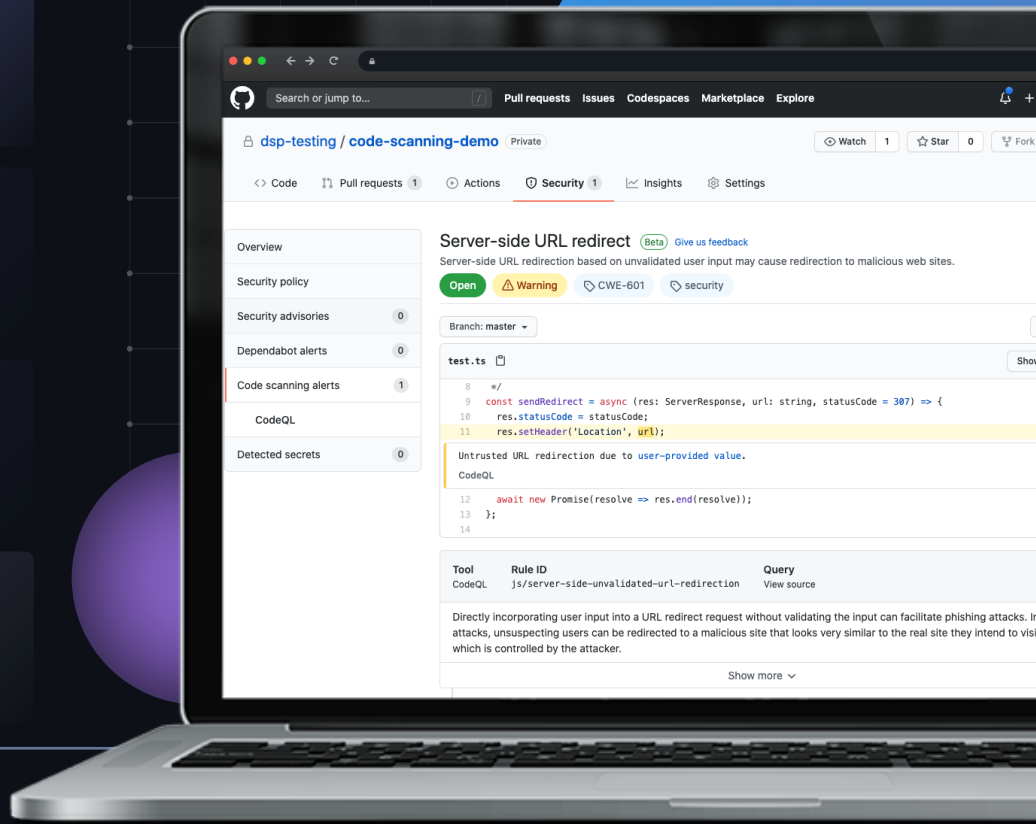
For every commit made to your repository and its full git history, we'll look for secret formats from secret scanning partners.

Define custom patterns

Scan for patterns that are internal to your organization across your repositories.

Supports both public and private repos

Secret scanning watches both public and private repos for potential secret vulnerabilities.

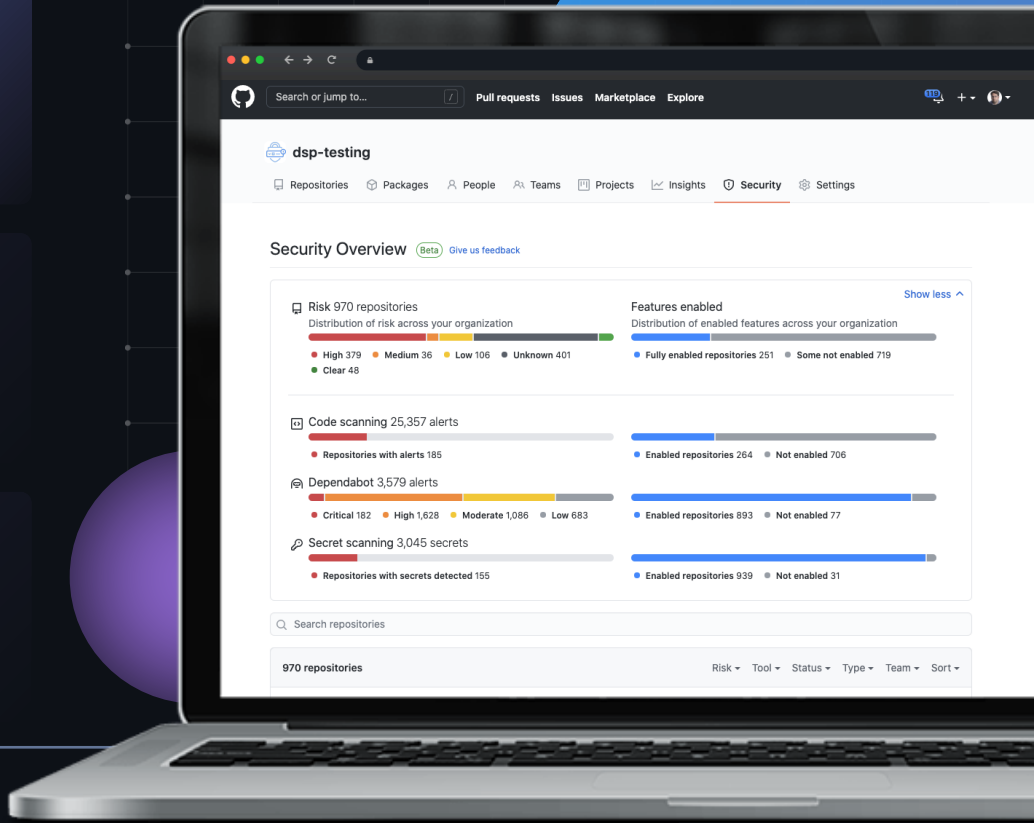


Security Overview

A single place to view code scanning, secret scanning and Dependabot alerts across your codebases

Determine the riskiest repos with high numbers of unaddressed alerts

Monitor enrollment status as you rollout security features across your organization



GitHub Advanced Security Lab

GitHub Advanced Security - Juice-Shop

- Navigate to https://github.com/juice-shop/juice-shop
- Fork the repo with your personal GitHub Account
- Click on the **Security** tab
- Click on **Dependabot** and enable(along with Dependency graph)
- Scroll down to **Code Scanning -> CodeQL Analysis -> Default**
- After waiting a few minutes, navigate back to Security and view Dependabot and Code scanning results.

GitHub Copilot

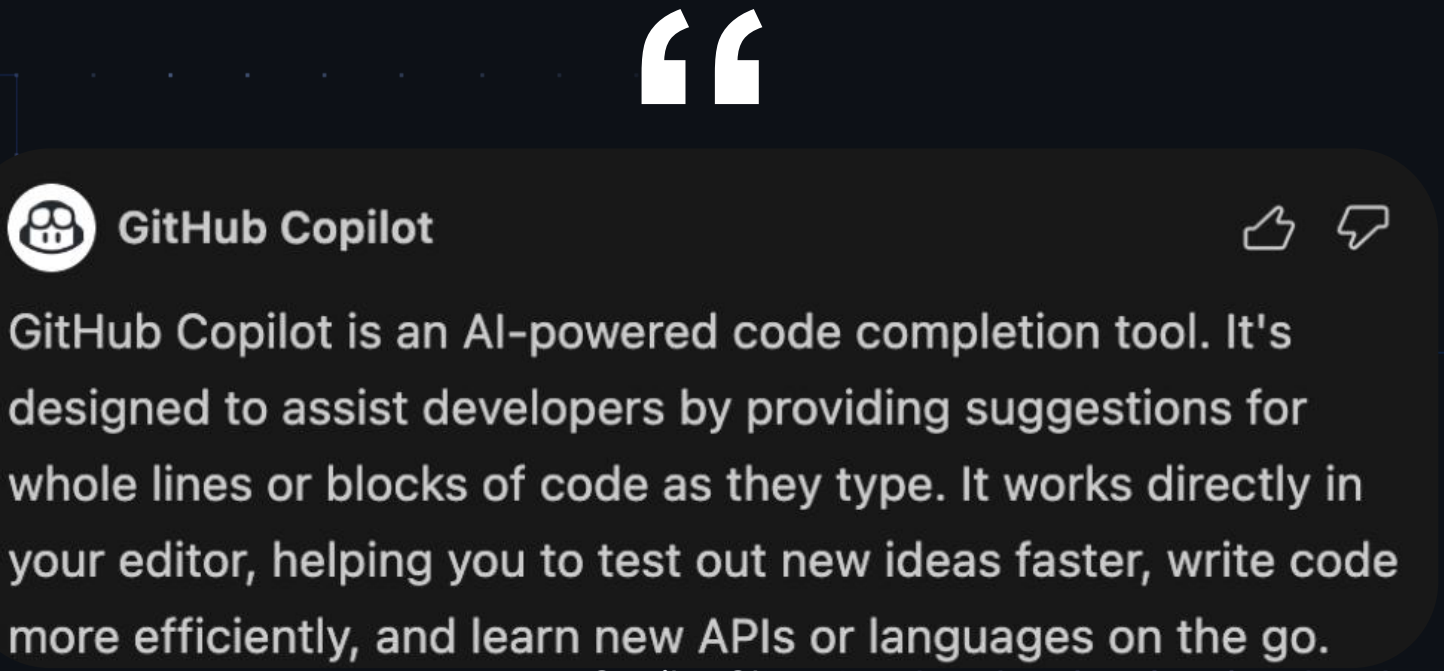
GitHub Copilot Fundamentals

What is GitHub Copilot?

What is required?

Supported IDEs

Data flow



- Copilot Chat

What is Required ?

GitHub account

Copilot license

Supported IDE

Supported IDEs

Visual Studio

Vim/Neovim

VS Code

Azure Data Studio

JetBrains IDE Suite

Xcode

How Copilot Fulfills a Request

Apply several pre-model filters

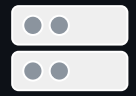
- Test for **toxic language**
- Test for **relevance** (chat only)
- Guard against **prompt hacking**

Copilot assembles context from tabs open in the code editor

OpenAI model

Contex / Prompt

N Suggestions



Proxy

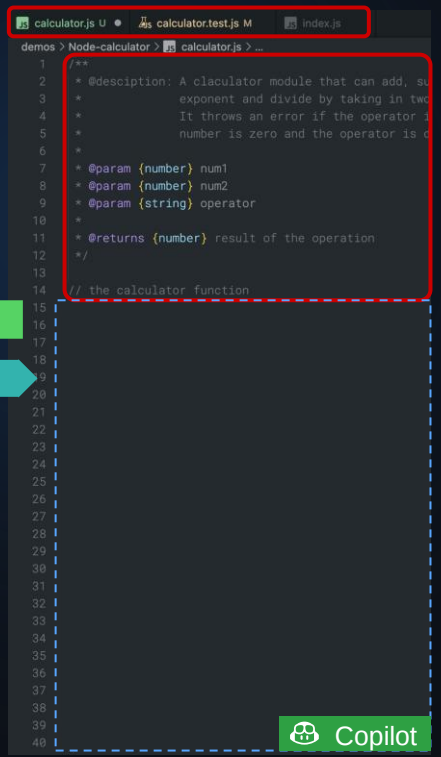
Contex / Prompt

N Suggestions

Copilot Agent

Contex / Prompt

N Suggestions



Apply several post-model filters

- **Code quality**
- **Unique Identifiers** (eg.: emails, IPs etc.)
- Suggestion **matching public code**

Feature overview

GitHub Copilot for Business - Features

GitHub Copilot for Enterprise - Features

GitHub Copilot - New Features

GitHub Copilot Roadmap

IDE Features

						
Code completion	✓	✓	✓	✓	✓	✓
Copilot Chat	✓	✓	✓			
Extensions	✓	✓	✓			
Knowledgebase	✓	✓				
Enhanced Skills	✓					
Code Review	✓					
Feature-rich	✓					
Custom Instructions	✓					

GitHub Copilot Business- Feature Overview

GA GA GA

Code completion

Code completion and suggestions

Copilot Chat

Receive answers to coding-related questions

Copilot CLI

Translate natural language into shell commands

GitHub Copilot Business- Feature Overview

GA GA GA

PR Summaries

Create summary of changes intro

Chat in the GUI

Search documentation directly from the UI

Content Exclusion

Exclude sensitive data to ensure secure AI code generation.

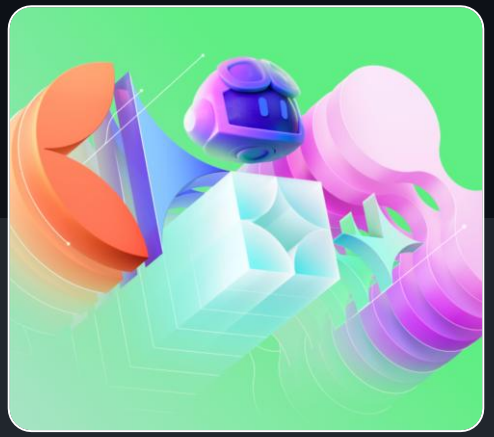
GitHub Copilot Business- Feature Overview

GA

GitHub Skills

Extend the functionality of Copilot Chat with Skills.

Public Preview



Model Picker

Bringing developer choice to Copilot with new models.

Public Preview

Private Extensions

Integrate your custom dev tools directly into Copilot Chat!

GitHub Copilot Enterprise- Feature Overview

Enterprise - GA

Enterprise - GA

Enterprise - GA

Knowledge bases

Leverage Organizational Context in Chat

Issues & Discussions

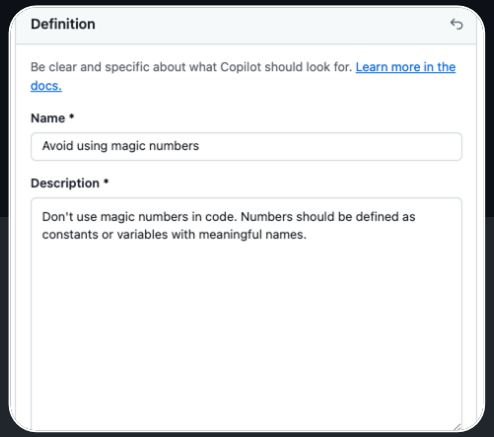
Understand context from your Issues & Discussions with Chat!

Enterprise Settings

Access to Enterprise Level Content Exclusion, Settings & Enterprise API Metrics

GitHub Copilot Enterprise- Feature Overview

Enterprise - GA



Coding Guidelines

Enhance Code consistency and quality with coding standards

Enterprise - Private Preview

Copilot Workspaces in PRs

Accelerate code reviews and improve collaboration within pull requests

Enterprise - Public Preview

Fine-Tuning

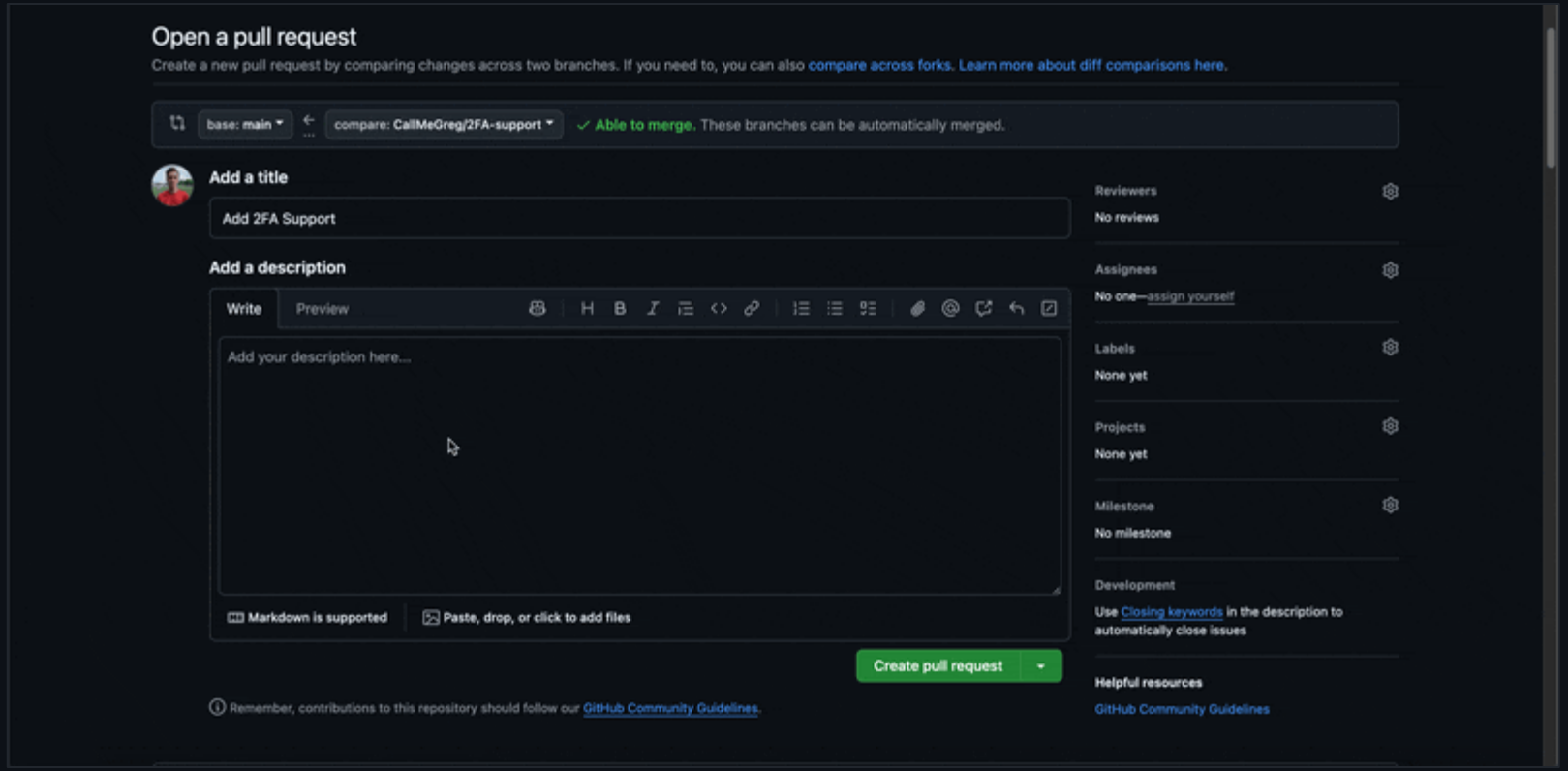
Train Copilot on your own data

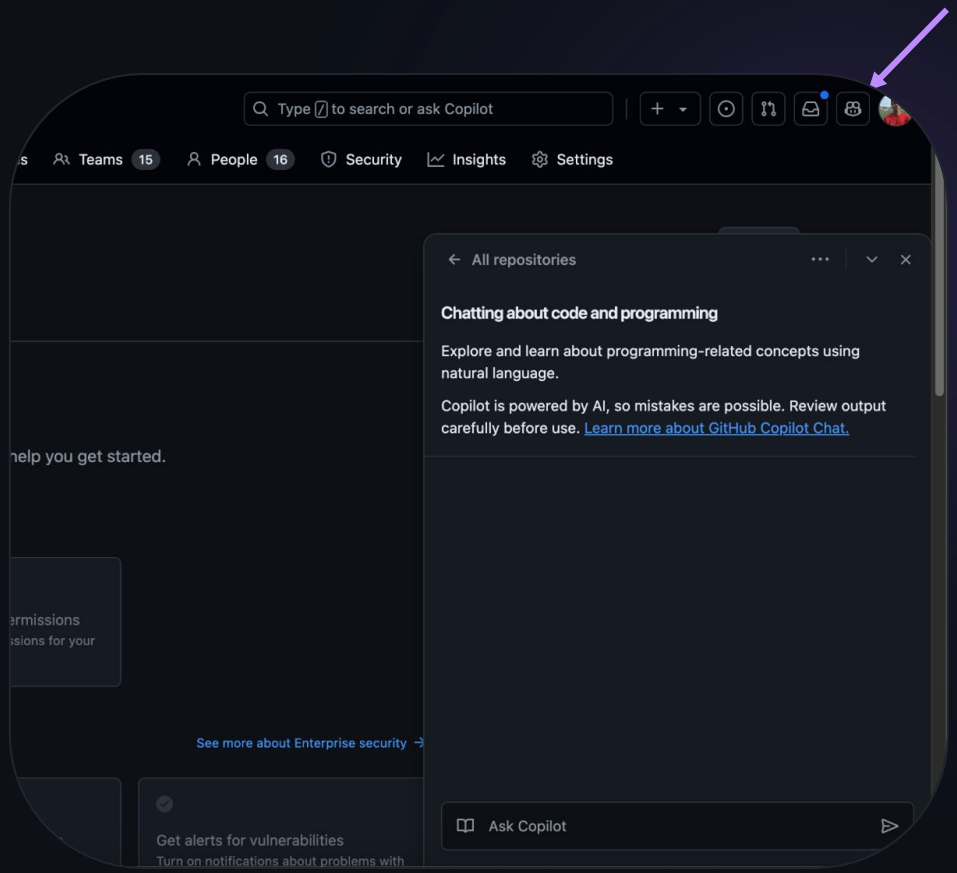
Recently Added Features

Pull Request (PR) Summaries

Copilot Chat in GitHub.com

Generating a pull request summary

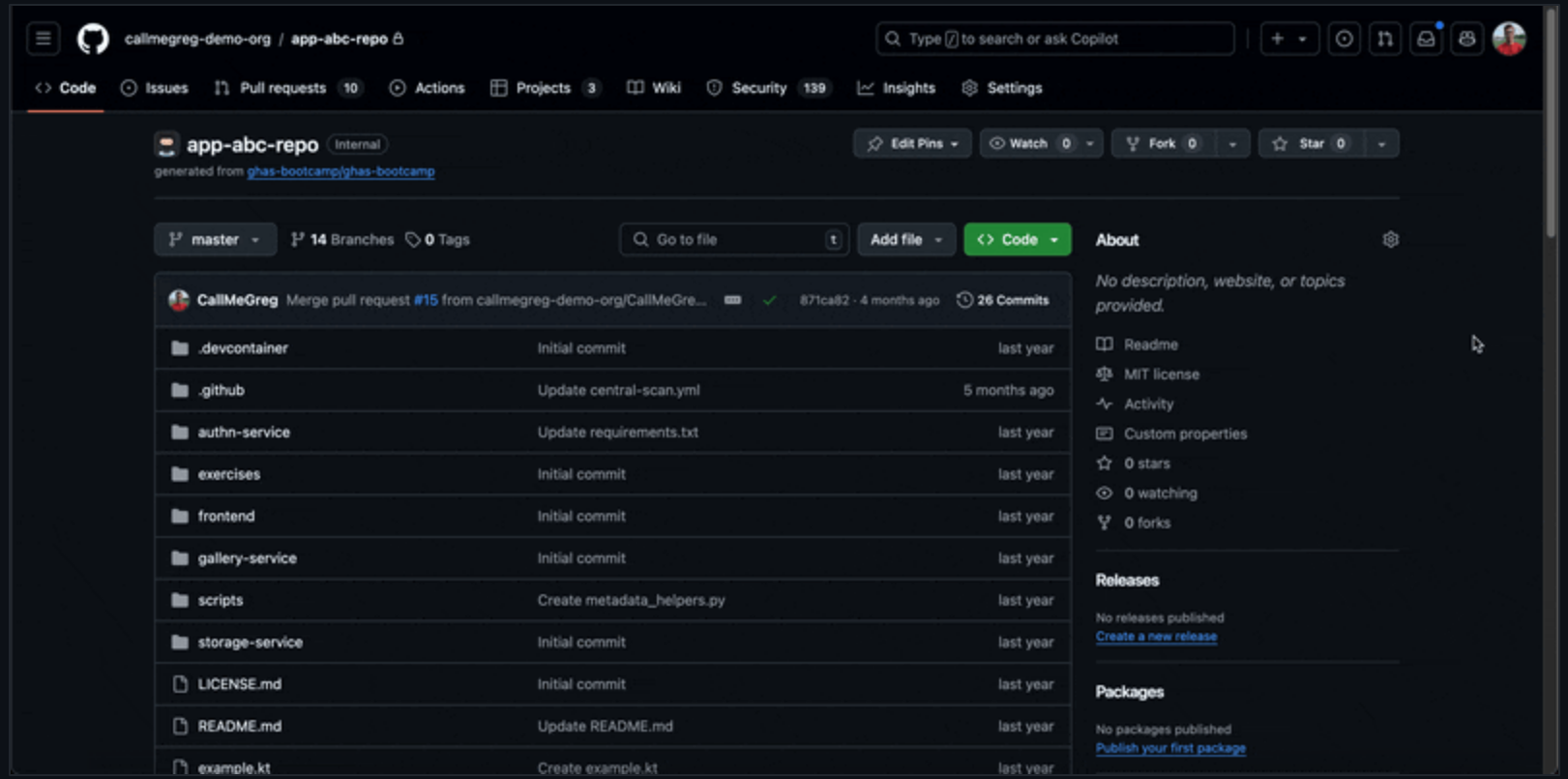




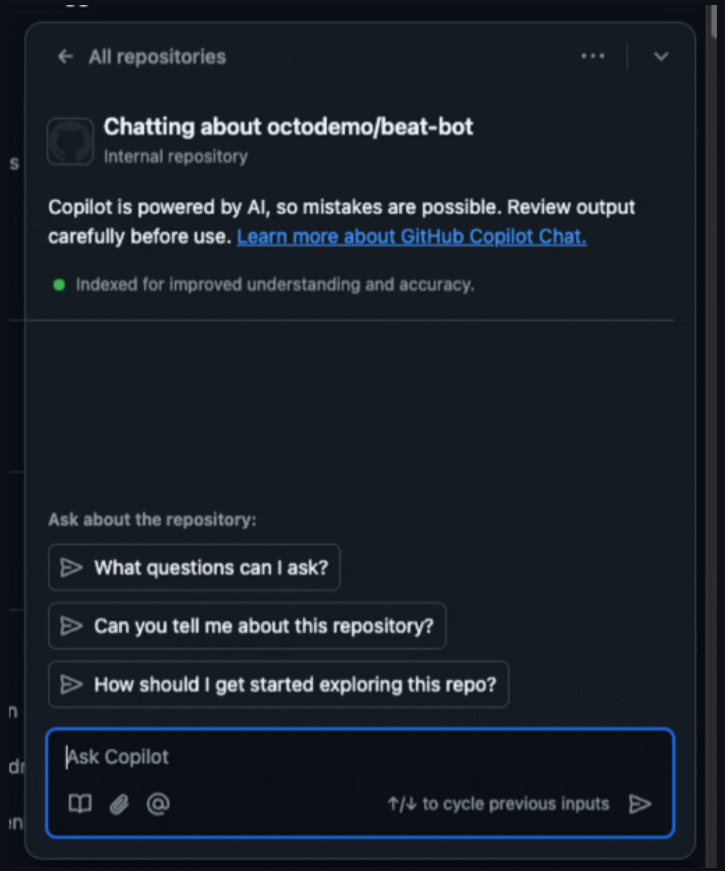
Copilot Chat in GitHub.com

- Index repositories to improve your suggestions
- Reference code, pull requests, issues (and more!) as context
- Search the web with less context switching

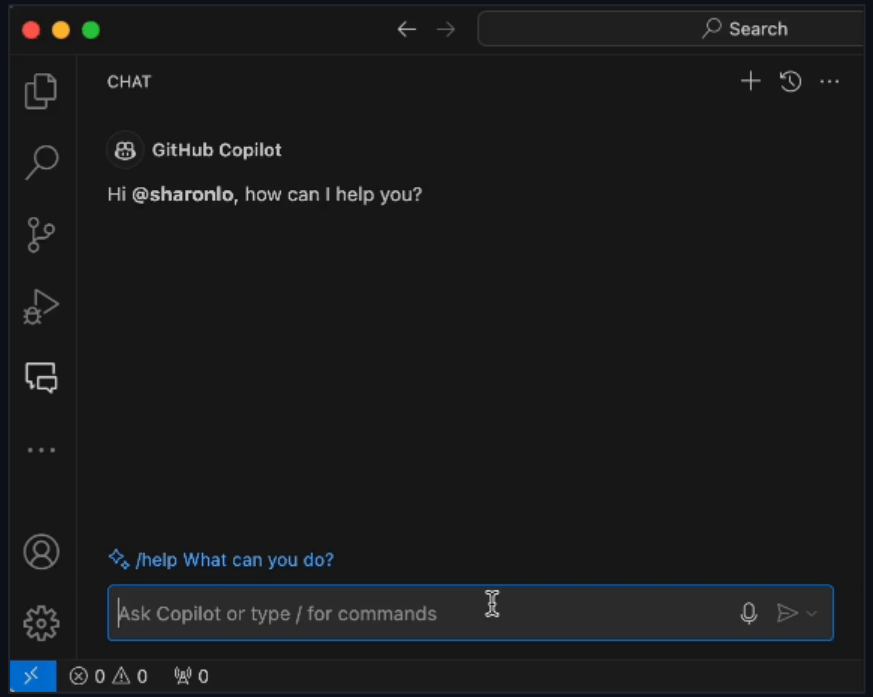
Indexing repositories



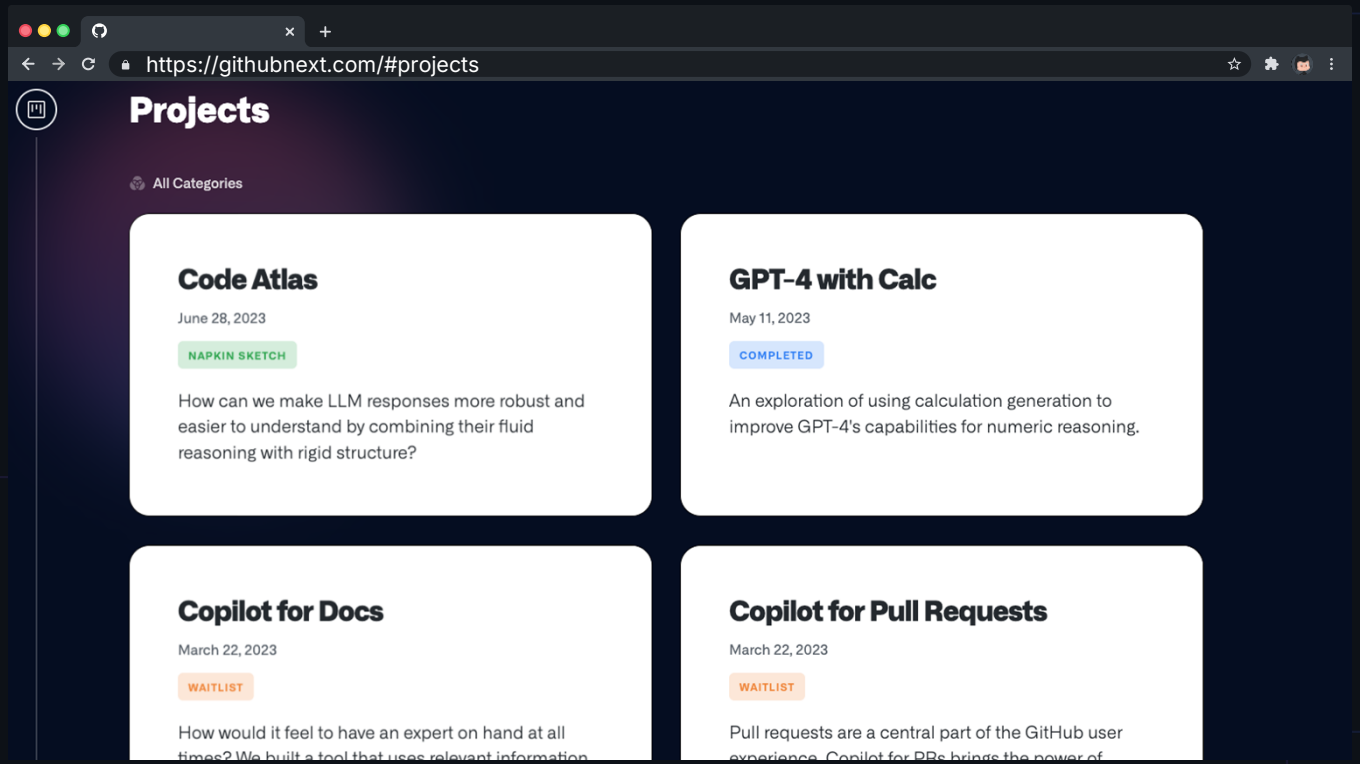
Copilot Chat is powered by “skills”



Search the web in real time with Bing



GitHub Copilot - What's coming up?



Note - This page is constantly updated. View https://githubnext.com/#projects for up to date info.

Coding

Copilot vs Copilot Chat

Copilot	Copilot Chat
Direct Code Writing	In-Depth Assistance
Seamless IDE Integration	Learning & Teaching
Solo Development	Collaborative Scenarios

Large Language Models (LLMs)
Limitations & caveats

GitHub Copilot leverages LLMs in order to produce output. Some general caveats related to LLMs should be observed when using GitHub Copilot or other LLM-based tools.

Outdated or Incomplete Training Data: LLMs are trained on historical data and may not account for newer language versions, frameworks, or libraries, leading to outdated or inaccurate suggestions (note: GitHub Copilot does have features to remedy this risk such as support for real-time online search).

Quality of Suggestions: Libraries, frameworks, and languages are not equally represented in the training data which may impact the quality of the suggestions.

Hallucinations: LLMs can produce output that are “hallucinations” i.e. responses presented as accurate but that are not. Always inspect the output! (more on how to remedy this later)

Using Copilot

GitHub Copilot

- Code Driven Development
- Multilingual
- Offering Alternative Results
- Create Unit Tests
- Documentation

Guide Copilot

Good code techniques

Use good names

GitHub Copilot understands natural language

Spell out variable names

Single letter variables and abbreviations are ambiguous

Keep functions functional

Follow strong principles when creating named blocks

Be consistent

Generated code follows contextual patterns

Helpful Patterns

Variable names

Use descriptive variable names to make your intentions clear.

```
total_attendees = 5
```

Method Signatures

Define method signatures with unambiguous parameter names and types.

```
calculateAverage(int num)
```

Naming Conventions

Maintain consistent naming conventions for variables and functions

i.e. using `camelCase` for variable names consistently

Helpful Patterns (Cont.)

Input/Output Format

Describe the expected input and output formats.

"Write a function that takes an array of strings as input and returns true if a palindrome is found."

Error Handling

Specify error handling scenarios

"Exit where the integer input is empty and throw an error if the input is not an integer at all"

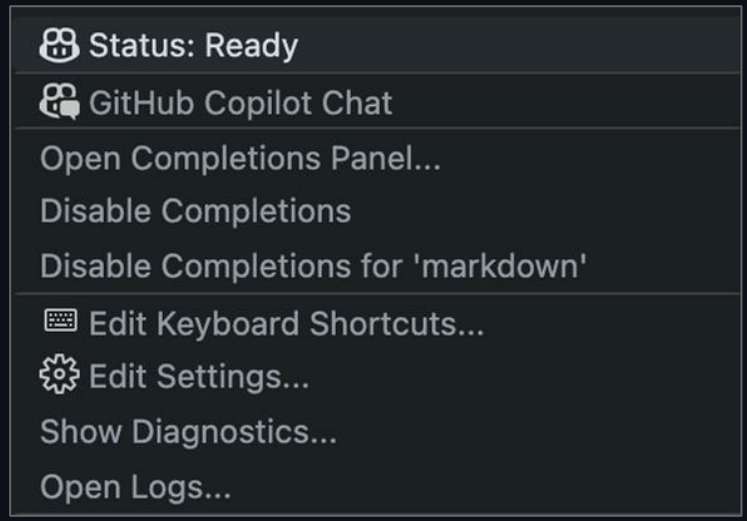
Control Structures

Describe control flow structures.

"Write a while loop to find the first prime number given a list of integers. If no such number exists, the loop should exit."

Advanced Settings

Configuration Options



Configuring GitHub Copilot in your environment

Using Copilot Chat

GitHub Copilot Chat

- Refactoring Code
- Generating Tests
- Debugging
- Creating a Workspace
- Documentation

Copilot Chat: Slash Commands

/help to find available commands in your IDE

In-file Copilot Chat

Copilot offers in-file Copilot feature to selectively improve

Run in Terminal

Ask in GitHub Copilot CLI

GitHub Copilot Chat (default)

Copilot Prompt Engineering

What

Prompt engineering is the process of designing and creating high-quality prompts that can be used to generate accurate and useful code suggestions with Copilot.

Why

- Maximizes utility of Copilot
- Minimizes repeated iterations
- Determines code suggestion quality
- Skill prompts guide Copilot to understand context & nuances
- Refined interactions reduces misunderstandings

How

- Neighboring Tabs
- Zero-Shot Prompting
- One-Shot Prompting
- Few-Shot Prompting
- Let's Think Step by Step

Prompting Best Practices

Improving results

These are best practices, now let’s see how those can be used in different ways

Provide references
Improve relevance of the response by providing an example and context

Write clear instructions
Refine your prompt, provide context, write clearly, and give Copilot ample input for better results

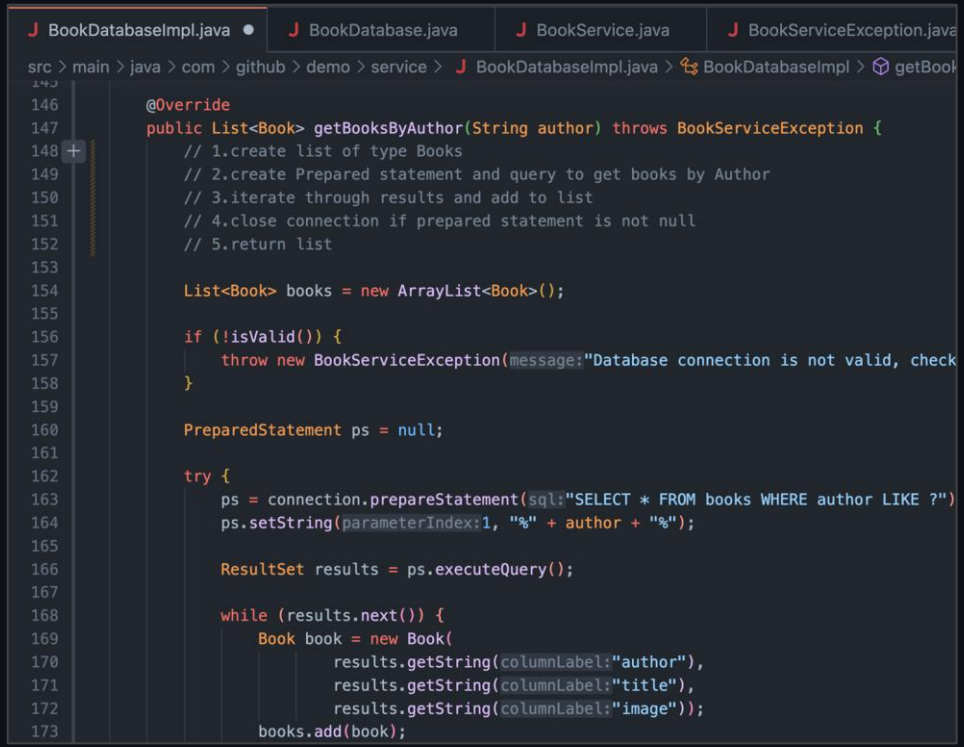
Split up big tasks
Breaking down complex tasks minimizes errors and utilizes previous outcomes for efficiency

Allow time to think
Requesting Copilot’s thought process will enhance Copilot accuracy, but it may prolong wait times.

Test changes systematically
Measure performance, watch for prompt changes' side effects, and use test suites for implementation

Neighboring Tabs

Keep relevant files open, things related to your current code



If at first you don’t succeed, iterate!

Zero Shot Prompt

```
//Write a JavaScript function that finds
//the maximum value in an array.
function max(array) {
    return Math.max.apply(null, array);
}
```

One-Shot Prompt

```
//Create a JavaScript function, `findMax`, that takes an array of
//and returns the string with the maximum length.
//Ensure that the function works correctly for arrays that are em
//as well as arrays that contain both strings and numbers.
function findMax(array) {
    var max = array[0].length;
    array.map(v => max = Math.max(max, v.length));
    result = array.filter(v => v.length == max);
    return result;
}
```

Provide a high level description to clarify expectations

calculations.py > ...

```python
"""
Create a Python function, `sum_array_values`, that finds
and returns the sum of integers in an array.

1.Verifiy if the array is empty and exit if it is.
2.Throw error if the array contains non-integer values.
3.Return the sum of the array values if the array is valid.

Please consider the following example data:
Example 1: [5, 10, 15, 20, 8]
Example 2: [2, 4, 6, 1, 9, 12]
The generated code should work correctly for arrays like above
"""
#define sum_array_values function
def sum_array_values(array):
    # 1.Verifiy if the array is empty and exit if it is.
    if len(array) == 0:
        return 0
    # 2.Throw error if the array contains non-integer values.
    for i in array:
        if type(i) != int:
            raise TypeError("The array contains non-integer values.")
    # 3.Return the sum of the array values if the array is valid.
    return sum(array)
```

Secure coding

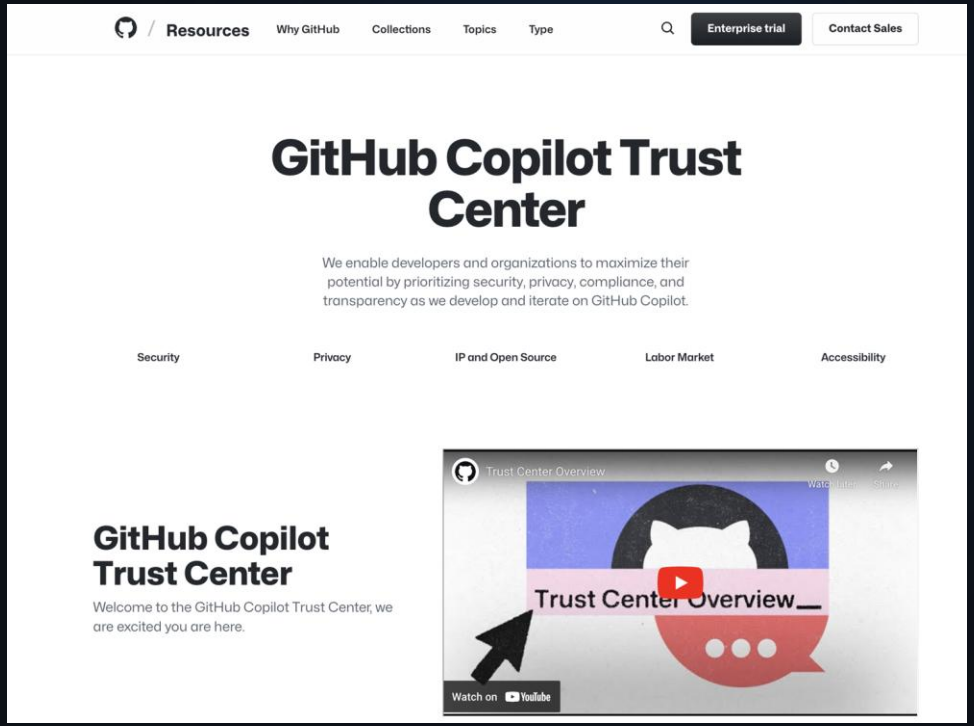
Copilot and Secure Coding

- AI-based vulnerability system that helps prevent insecure coding patterns (e.g. SQL script injection)
- Vulnerability filters are applied only to the Copilot-generated suggestion itself
  - ⊗ It cannot detect downstream vulnerabilities introduced by the code e.g. on deployment infrastructure
  - ⓘ **We recommend taking the same precautions you take with code written by engineers (linting, code scanning, etc.)**
- Copilot Chat can be used to query code for known vulnerabilities

Security & Trust

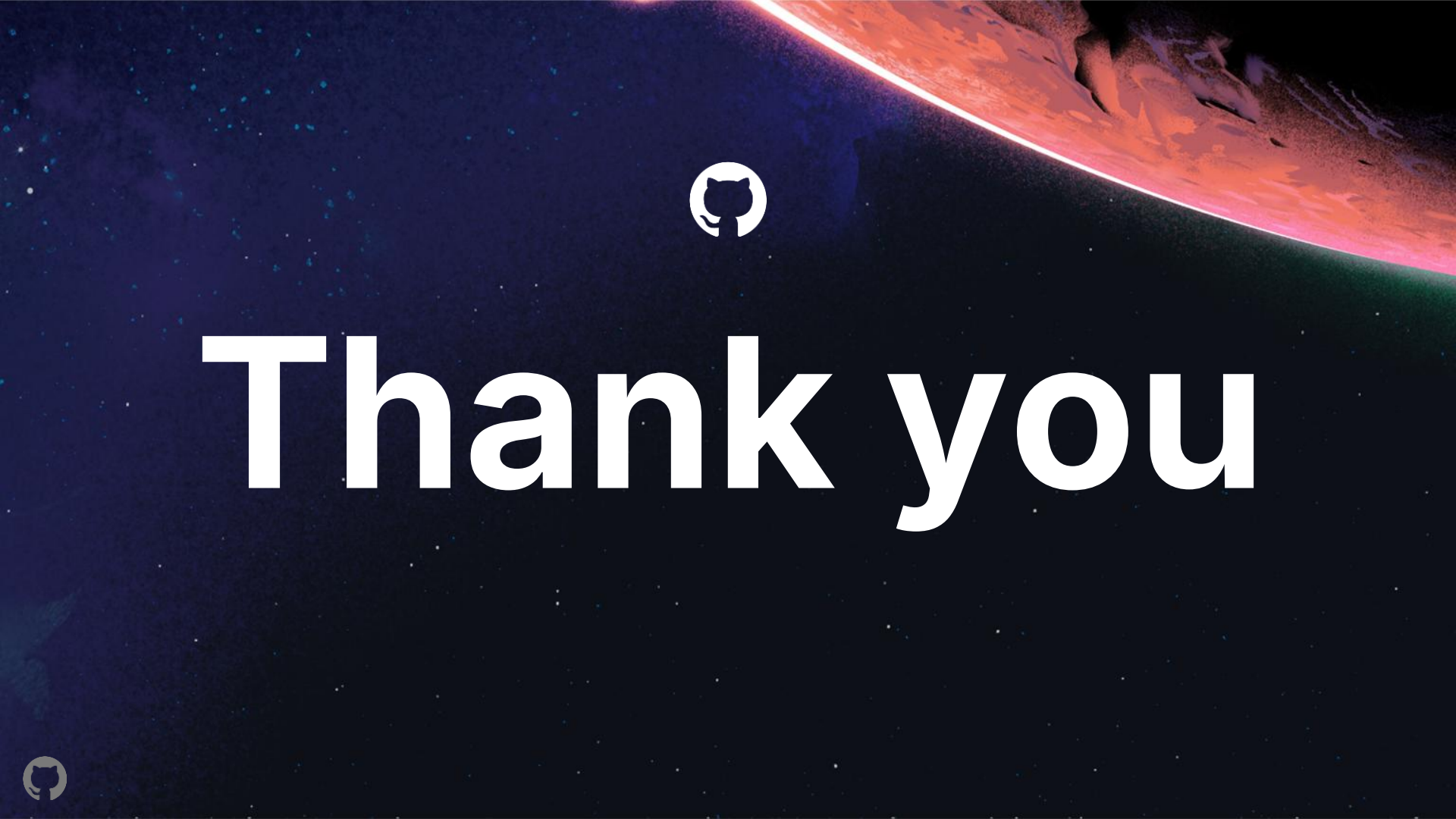
Copilot Trust Center

- Security
- Privacy
- Data flow
- Copyright
- Labor market
- Accessibility
- Contracting



GitHub Copilot Lab – Copilot Adventure

- https://microsoft.github.io/CopilotAdventures/
- Challenges can be done in C#, JS, or Python
- Start with the Warmup Adventure
- Select one of the beginner adventures
- Use either GH Codespaces or locally with VS Code


Thank you