GitHub Administration

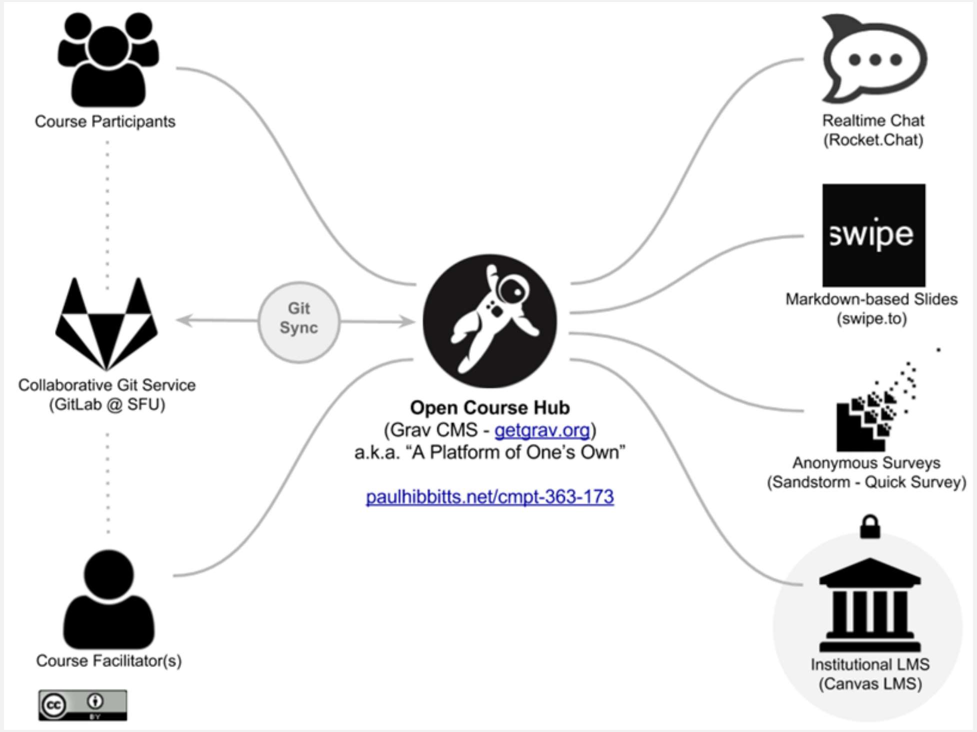
1. Getting Started with Collaboration

What is GitHub?

- People use GitHub to build some of the most advanced technologies in the world.
- Whether you’re visualizing data or building a new game, there’s a whole community and set of tools on GitHub that can help you do it even better.

GitHub is a collaboration platform that uses Git for versioning. GitHub is a popular place to share and contribute to open-source software.

The GitHub Ecosystem



What is Git?

- Git is a free and open source distributed code management and Version control system that is distributed under the GNU General Public License version 2.

- In addition to software version control, Git is used for other applications including configuration management and content management.

Git Features

- Distributed version control
- Atomic commits
- Strong support for non-linear development
- Efficient handling of large projects
- Cryptographic authentication of history
- Efficient branching and tagging
- Toolkit design

Exploring a GitHub Repository

- A repository is the most basic element of GitHub, They're easiest to imagine as a project's folder.

- A repository contains all of the project files (including documentation), and stores each file's revision history.

- Repositories can have multiple collaborators and can be either public or private.

Using GitHub Issues

- Issues let you track your work on GitHub, where development happens.

- When you mention an issue in another issue or pull request, the issue's timeline reflects the cross-reference so that you can keep track of related work.

Quickly create issues

- Issues can be created in a variety of ways, so you can choose the most convenient method for your workflow.

- For example, you can create an issue from a repository, an item in a task list, a note in a project, a comment in an issue or pull request, a specific line of code, or a URL query.

2. Getting Started with GitHub

Finding and Evaluating Repositories

- There various techniques to efficiently search GitHub, enabling you to find specific issues, pull requests, repositories, users, topics, and more..

- GitHub Search Queries

Search Issues and Pull Requests on GitHub

- is:issue is:open label:beginner - This particular query will list all projects with issues that are open and labeled beginner.
- is:issue is:open label:easy - This will list all open issues that are labeled easy.
- is:issue is:open label:first-timers-only - This lists all open issues that welcome first-timer contributions.

Search Repositories

- Finding by Name, Description, README, or Topic
- Combining Queries
- Finding by Stars and Forks
- Finding by Language
- Finding by Organization Name
- Finding by Date
- Finding by License
- Finding by Visibility

Search Repositories - In

- Using in:name. Let's say you are looking for resources to learn more about Data Science. In this case, you can use the command Data Science in:name which will list repositories with Data Science in the repository name.

- Using in:description. If you want to find repositories with a ceratin description

- Using in:readme. You use this to search through a README of a file for a certain phrase.

Search Repositories- Stars, Forks.

Using stars:n. If you search for a repository with 1000 stars, then your search query will be stars:1000. This will list repositories with exactly 1000 stars.

Using forks:n. This specifies the number of forks a repository should have. If you want find repositories that have less than 100 forks, your search will be: forks:<100.

Search Repositories- Language, Organization Name

Using language:LANGUAGE. For example if you want to find repositories written in PHP, your search will be: language:PHP

org:linux it will list repositories that match linux.

Search Repositories- Date, License.

Using keyword:YYYY-MM-DD. Take an instance where we want to make a search of all repositories with the word Nvidia that were created after 2022-10-01. Then our search will be: Nvidia created:>2022-10-01

Using license:LICENSE_KEYWORD. This is a good way to search for projects with specific licenses. To search projects with the MIT license, for instance, you would do license:MIT.

Search Repositories- Visibility.

Using is:public. This will show a list of public repositories. Let's take an instance where we want to search all public repositories owned by Google. Then our search will be: is:public org:google.

Using is:private. This query is meant to lists all private repositories under the given search query.

Evaluating Repositories

- Stars: Indicator of popularity and community interest
- Forks: Shows how many people have copied the repository to contribute or modify
- Recent activity: Check for frequent commits, issue updates, and pull request activity to gauge project health

Checking for a Well-Maintained README and Active Issues

A README file is often the first point of contact for anyone interacting with a repository. It serves several critical functions:

- Overview
- Setup Instructions
- Usage
- Contribution Guidelines

Checking for a Well-Maintained README and Active Issues

Active Issues: Sign of an Engaged Community and Ongoing Development

- Engaged Community
- Ongoing Development
- Quality Assurance
- Transparency

Using GitHub Explore to Discover Projects

Access via the GitHub homepage or directly at github.com/explore.

Customizing Recommendations:
•Following topics and developers to tailor the Explore feed
•Using tags and trending repositories to find relevant projects
•Keeping up with popular and emerging projects

Following Trending Projects and Developers:
•Engaging with influential developers in your field

Signing up for a GitHub account

Creating and Configuring Your User Account

- Signing up for a GitHub account
- Configuring email, username, and password
- Setting up two-factor authentication for security

Authenticating to GitHub

Creating and Configuring Your User Account

- Using SSH keys for secure access
- Personal access tokens for API and command-line access

Setting Up and Managing Your GitHub Profile

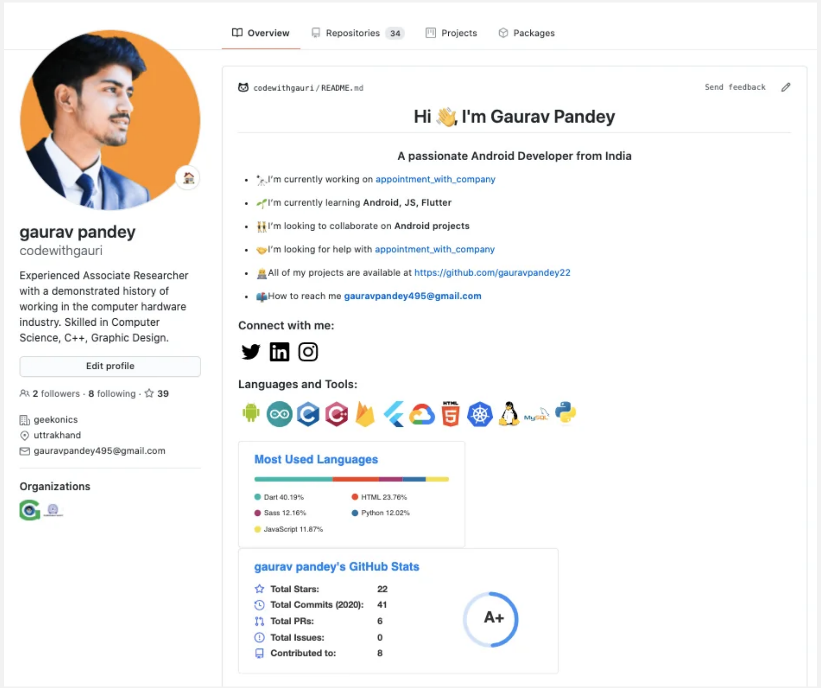
- Personalizing Your Profile Page
- Adding a profile picture and cover image
- Customizing your bio with key information
- Pinning repositories to showcase

- Writing an Effective Bio and Showcasing Your Work

Writing on GitHub

Markdown Basics for GitHub

- What is Markdown syntax?
- Formatting text, lists, and links
- Adding images and code blocks



gaurav pandey
codewithgauri

Experienced Associate Researcher with a demonstrated history of working in the computer hardware industry. Skilled in Computer Science, C++, Graphic Design.

2 followers · 8 following · 39

geekonics
uttrakhand
gauravpandey495@gmail.com

Organizations

codewithgauri / README.md

Hi 👋, I'm Gaurav Pandey

A passionate Android Developer from India

- 🔭 I'm currently working on appointment_with_company
- 🌱 I'm currently learning **Android, JS, Flutter**
- 👯 I'm looking to collaborate on **Android projects**
- 🤝 I'm looking for help with appointment_with_company
- 👨‍💻 All of my projects are available at https://github.com/gauravpandey22
- 📫 How to reach me **gauravpandey495@gmail.com**

Connect with me:

Languages and Tools:

Most Used Languages

● Dart 40.19% ● HTML 23.76%
● Sass 12.16% ● Python 12.02%
● JavaScript 11.87%

gaurav pandey's GitHub Stats

Total Stars:	22
Total Commits (2020):	41
Total PRs:	6
Total Issues:	0
Contributed to:	8

A+

COMPLETE LAB

3. Communication and Collaboration Tools

Receiving Notifications About Activity on GitHub

Setting Notification Preferences

Global Settings: apply to your entire GitHub account and influence how you receive notifications for activities across all repositories you are involved with.

Repository-Specific Settings: allow you to customize notifications for individual repositories. This is useful when you want to receive different types or frequencies of notifications for different projects.

Managing Email and Web Notifications

Managing Email Notifications:

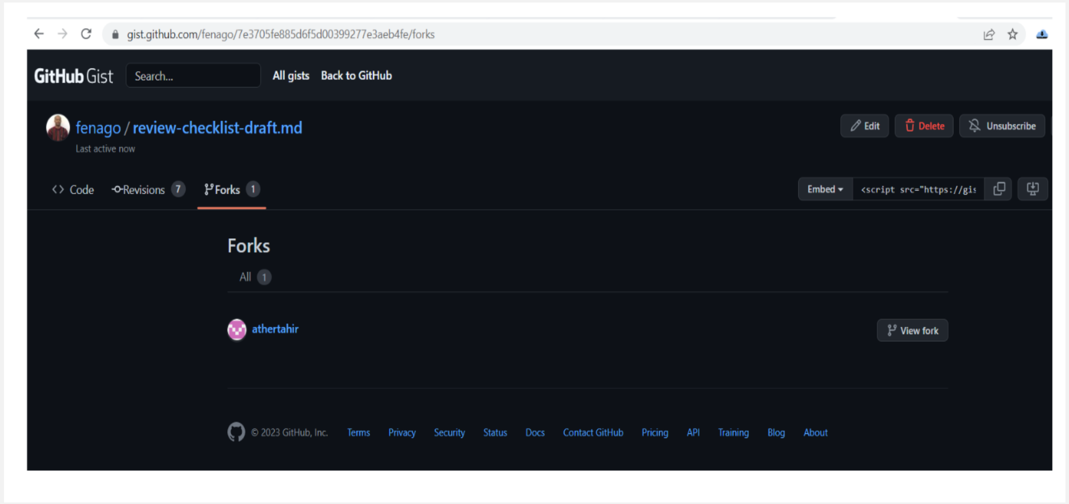
- Customize the frequency and type of emails you receive (e.g., immediate, daily, or weekly digests).
- Filter GitHub emails into specific folders using email rules or filters.

Managing Web Notifications:

- Access notifications by clicking the bell icon in the upper-right corner of GitHub.
- Organize notifications by marking them as read, unsubscribing from threads, or muting conversations.

Working with Gists

Gists are a feature provided by GitHub to easily share and manage code snippets, notes, or any other type of text content. They are particularly useful for sharing small pieces of code or configuration that don't require a full-fledged repository. Gists can be public or secret, allowing for flexible sharing options.



Using the GitHub Wiki

A GitHub Wiki is a feature provided by GitHub to help developers document their projects. It's a place where you can create and manage documentation directly within your repository, making it easier for users to understand, use, and contribute to the project. The Wiki can be used for a variety of purposes, such as user guides, API documentation, developer notes, and more.

Best Practices for Documentation

The objective is to create high-quality documentation that is clear, concise, and useful. Good documentation enhances the user experience, facilitates collaboration, and makes it easier for new contributors to get involved. Regularly updating and maintaining documentation ensures it remains relevant and accurate, ultimately contributing to the success of your project.

Best Practices for Documentation

What are the best practices for documentation in GitHub:

- Clear and Concise Writing
- Consistent Formatting
- Contributing
- Advanced Usage
- API Documentation
- Troubleshooting
- Regular Updates
- Accessibility

GitHub Pages Basics

Key Features of GitHub Pages:

- Static Site Hosting
- Custom Domains
- Jekyll Integration
- Automatic Deployment

Types of GitHub Pages:

- User or Organization Pages
- Project Pages

GitHub Pages Basics

GitHub Pages offers a variety of customization options to create a unique and personalized website. Here are the key ways to customize your GitHub Pages site:

- Using Themes
- Customizing Layouts and Styles
- Adding Custom Content with Markdown
- Using Jekyll for Advanced Customization
- Using Plugins
- Setting Up a Custom Domain
- Adding Analytics

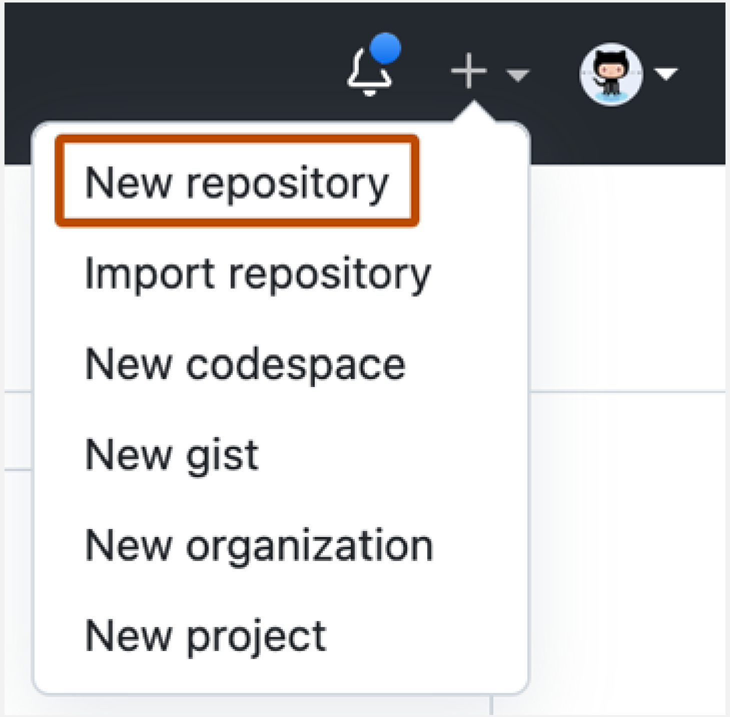
COMPLETE LAB 2

4. Repository Management

Introduction to Repository Management

Key Aspects of Repository Management:

- Creating, Cloning, and Archiving Repositories
- Managing Files in a Repository
- Managing Remotes



Introduction to Repository Management

Benefits of Effective Repository Management

Collaboration: Enhances teamwork by providing clear structure and version control.

Maintenance: Simplifies code updates, bug fixes, and feature additions.

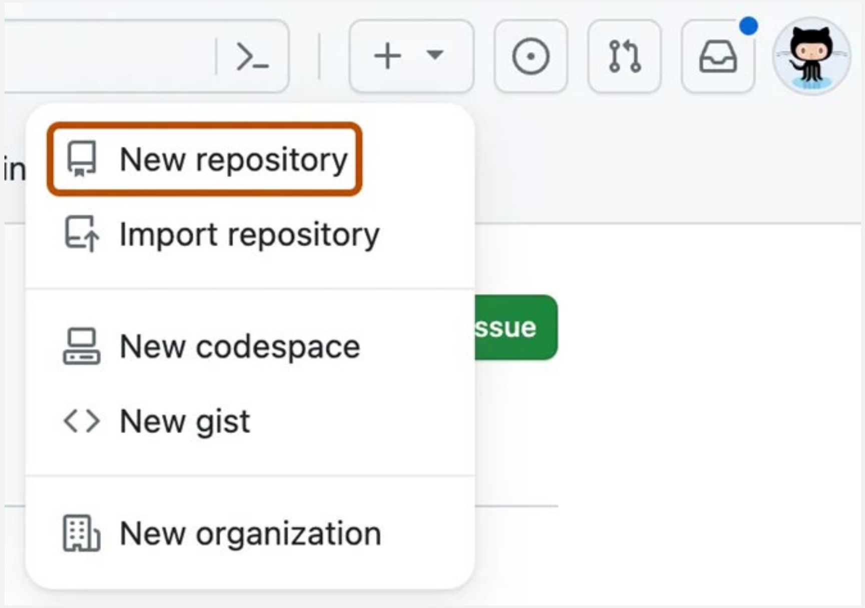
History Tracking: Keeps a detailed history of changes for auditing and rollback.

Efficiency: Streamlines development workflows and deployment processes.

Creating, Cloning, and Archiving Repositories on GitHub

Creating a Repository:

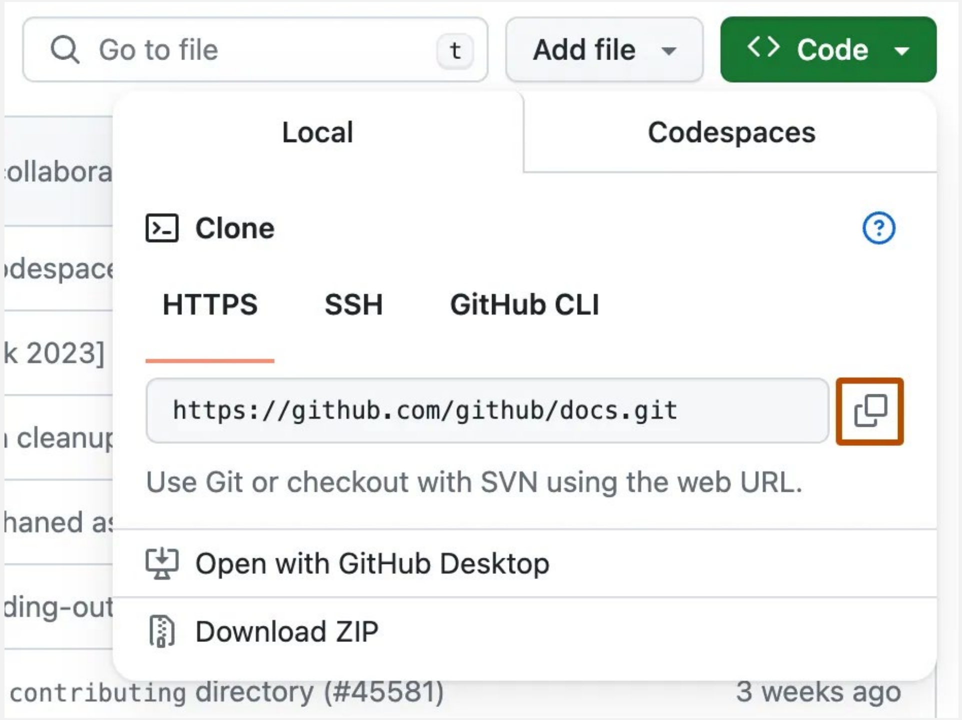
Purpose: To start a new project or centralize code for a specific purpose.



Creating, Cloning, and Archiving Repositories on GitHub

Cloning a Repository:

Purpose: To create a local copy of a repository for development.



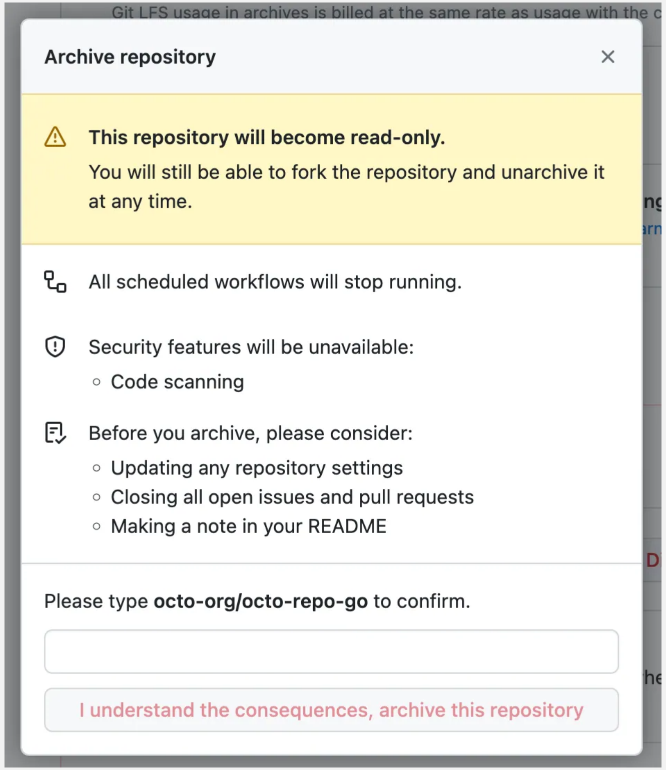
```
https://github.com/github/docs.git
```

Use Git or checkout with SVN using the web URL.

Creating, Cloning, and Archiving Repositories on GitHub

Archiving a Repository:

Purpose: To mark a repository as read-only, preserving its content without allowing further changes.


Git LFS usage in archives is billed at the same rate as usage with the c

Archive repository

This repository will become read-only.
You will still be able to fork the repository and unarchive it at any time.

All scheduled workflows will stop running.

Security features will be unavailable:
- Code scanning

Before you archive, please consider:
- Updating any repository settings
- Closing all open issues and pull requests
- Making a note in your README

Please type **octo-org/octo-repo-go** to confirm.

Managing Files in a Repository

Key Tasks in Managing Files

- Adding Files
- Removing Files
- Editing Files
- Working with Non-Code Files
- Managing Large Files with Git LFS

Managing Files in a Repository

Key Tasks in Managing Files

- Adding Files
- Removing Files
- Editing Files
- Working with Non-Code Files
- Managing Large Files with Git LFS

Managing Files in a Repository

Key Tasks in Managing Files

- Adding Files
- Removing Files
- Editing Files
- Working with Non-Code Files
- Managing Large Files with Git LFS

Managing Files in a Repository

Key Tasks in Managing Files

- Adding Files
- Removing Files
- Editing Files
- Working with Non-Code Files
- Managing Large Files with Git LFS

Managing Files in a Repository

Key Tasks in Managing Files

- Adding Files
- Removing Files
- Editing Files
- Working with Non-Code Files
- Managing Large Files with Git LFS

Managing Remotes

A remote URL is Git's fancy way of saying "the place where your code is stored." That URL could be your repository on GitHub, or another user's fork, or even on a completely different server.

You can only push to two types of URL addresses:

1. An HTTPS URL like https://github.com/user/repo.git
2. An SSH URL, like git@github.com:user/repo.git

Git associates a remote URL with a name, and your default remote is usually called origin.

Adding Remote Repositories

Adding a remote repository links your local repository to a remote version, allowing you to push and pull changes.

```
git remote add origin https://github.com/username/repository.git
```

Removing Remote Repositories

Removing a remote repository disconnects your local repository from the remote version.

```
git remote remove origin
```

Fetching and Pushing Changes to Remotes

Fetching updates your local repository with changes from the remote repository without merging them into your working directory. This allows you to review the changes before integrating them.

```
git fetch origin
```

OUTPUT

```
remote: Enumerating objects: 5, done.remote: Counting objects: 100% (5/5),
done.remote: Compressing objects: 100% (3/3), done.remote: Total 3 (delta
1), reused 0 (delta 0), pack-reused 0Unpacking objects: 100% (3/3), 1.23 KiB |
629.00 KiB/s, done. From https://github.com/username/repository
a1b2c3d..d4e5f6g  main       -> origin/main
```

Fetching and Pushing Changes to Remotes

Pushing sends your local commits to the remote repository. This updates the remote repository with your local changes.

```
git push origin main
```

OUTPUT:

```
Enumerating objects: 5, done.Counting objects: 100% (5/5), done.Delta compression using up to 4 threadsCompressing objects: 100% (3/3), done.Writing objects: 100% (3/3), 370 bytes | 370.00 KiB/s, done.Total 3 (delta 1), reused 0 (delta 0)To https://github.com/username/repository.git   a1b2c3d..d4e5f6g main -> main
```

5. Advanced Repository Operations

Administering a Repository

Setting repository visibility

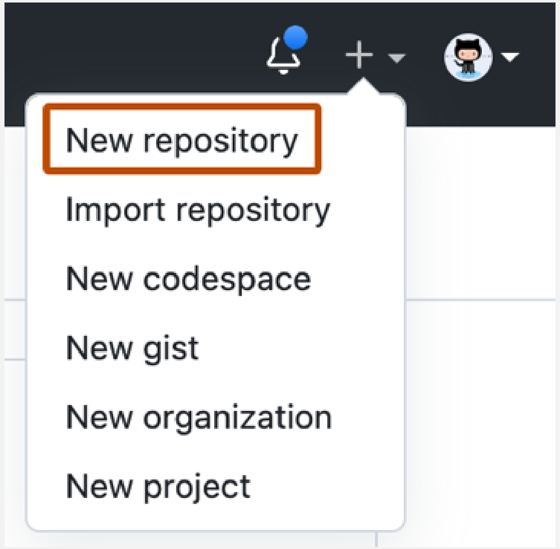
Types of Repository Visibility

Public Repositories:

- Visibility: Publicly accessible to anyone on the internet.

Private Repositories:

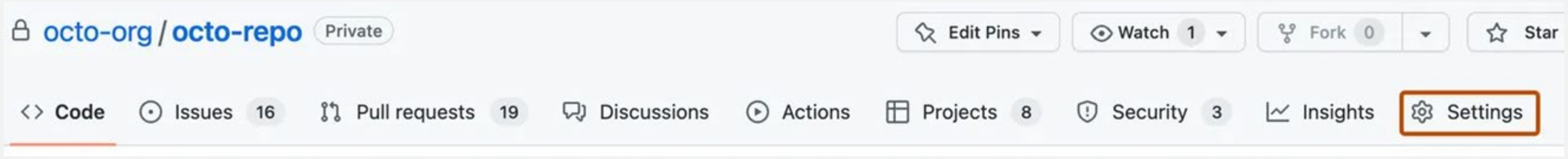
- Visibility: Accessible only to you and those you explicitly share it with.



Administering a Repository

Changing Repository Visibility

Go to Settings.
Scroll to "Danger Zone".
Click "Change visibility".



Administering a Repository

Collaboration Settings

Key Aspects of Collaboration Settings

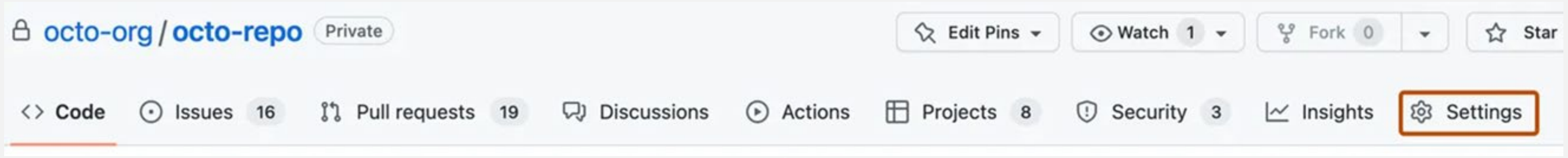
- Adding Collaborators
- Assigning Roles and Permissions
- Managing Team Access
- Branch Protection Rules

Administering a Repository

Adding Collaborators

Steps to Add Collaborators:

1. Navigate to the Repository:
2. Access Repository Settings:
3. Manage Access:
4. Invite Collaborators:
5. Confirm Invitation:



Administering a Repository

Assigning Roles and Permissions

GitHub provides different roles to control the level of access collaborators have.

The main roles are:

Read: Can view and clone the repository but cannot make changes.
Write: Can view, clone, and push changes to the repository.
Admin: Can view, clone, push changes, and manage settings for the repository.

Administering a Repository

Managing Team Access

For organizations, managing team access is more efficient than adding individual collaborators.

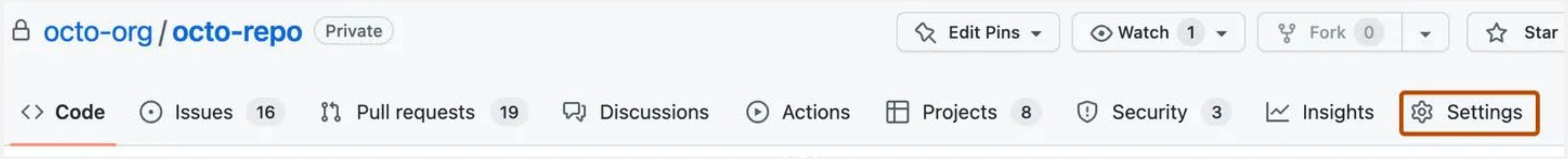
Teams can be granted varying levels of access to repositories, making it easier to manage permissions for groups of users.

Administering a Repository

Branch Protection Rules

Key Branch Protection Features:

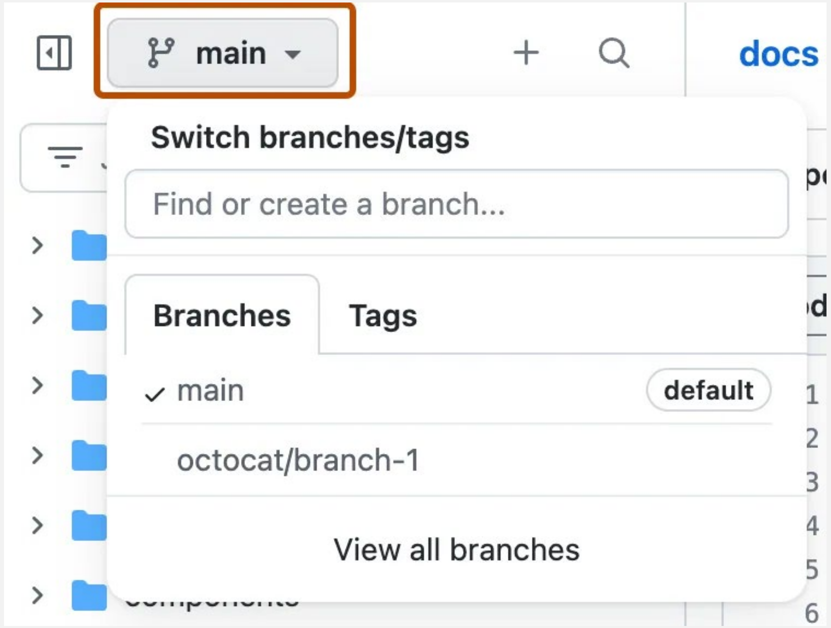
- Require pull request reviews before merging
- Require status checks to pass before merging
- Require signed commits: Require linear history
- Restrict who can push to matching branches



Administering a Repository

Managing Branches

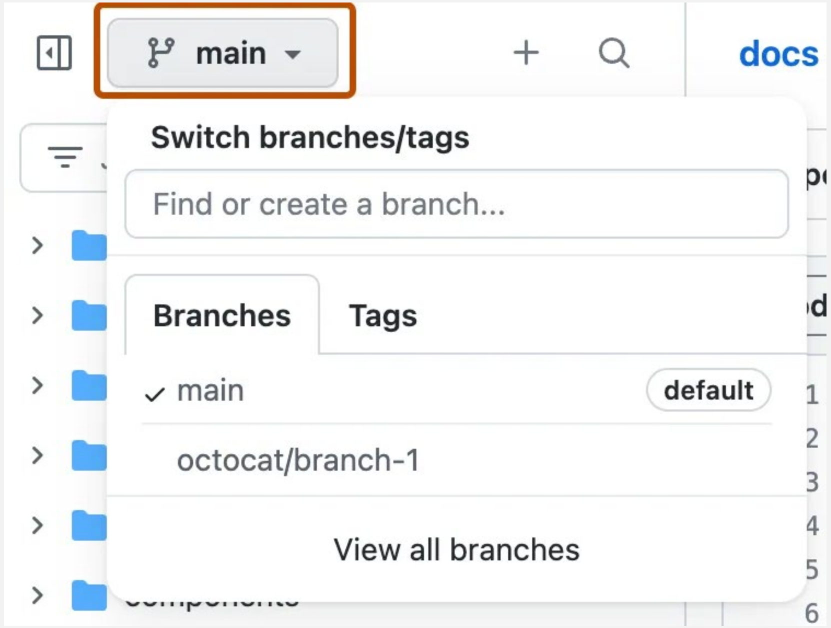
What is a Branch? A branch in Git is a separate line of development. By default, a repository has a branch named main (or master), which is considered the main line of development. Branching allows multiple developers to work on different features or fixes simultaneously without interfering with each other's work.



Administering a Repository

Creating / Manage Branches

From the file tree view on the left, select the branch dropdown menu, then click View all branches. You can also find the branch dropdown menu at the top of the integrated file editor.

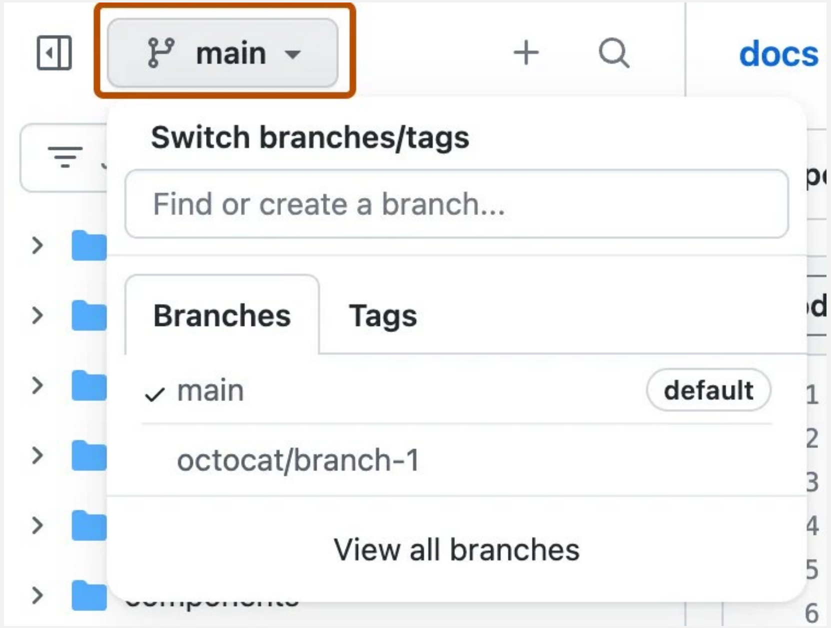


Administering a Repository

Managing Tags

A tag in Git is a reference to a specific point in the repository's history. Tags are often used to mark release points (e.g., v1.0, v2.0).

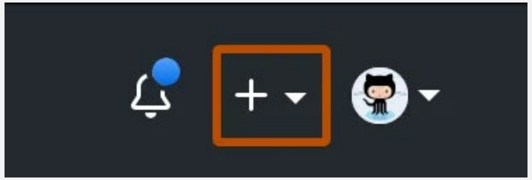
Unlike branches, tags are not meant to change.



Importing Your Projects to GitHub

GitHub Importer

GitHub Importer is a tool that quickly imports Git repositories from other hosting services to GitHub.



- In the upper-right corner of any page, click , and then click Import repository.

Importing Your Projects to GitHub

Other Tools to Import Your Projects into GitHub

- Git SVNGit: SVN is a Git command that allows you to interact with Subversion (SVN) repositories. This tool is useful for migrating SVN repositories to Git.
- Hg-Git: Hg-Git is a Mercurial extension that allows you to work with Git repositories from Mercurial, making it possible to convert Mercurial (Hg) repositories to Git.
- Fast-Import: Fast-Import is a fast-import and fast-export tool for Git. It allows importing projects from various VCS systems into Git.

Importing Your Projects to GitHub

Other Tools to Import Your Projects into GitHub

- Bitbucket to GitHub Migration: Bitbucket provides tools for migrating repositories to GitHub, especially useful for teams transitioning from Atlassian's Bitbucket.

- Third-Party Migration Services
  - GitKraken: GitKraken is a Git client that supports repository migration.
  - SVN2Git: A tool specifically for converting SVN repositories to Git.
  - Atlassian Migration Tools: Tools provided by Atlassian for migrating from Bitbucket and other Atlassian products.

Importing Your Projects to GitHub

Migrating from other version control systems

General Steps for Migration

1. Assessment and Preparation
   - Evaluate the Current VCS:
   - Choose Migration Tools:
   - Backup Data

2. Set Up Git Environment:
   - Install Git
   - Create a GitHub Repository

Importing Your Projects to GitHub

Migrating from other version control systems

4. Perform the Migration:
   - Use Migration Tools
   - Verify the Migration

5. Post-Migration Steps:
   - Update Remote URLs
   - Test the Repository
   - Document the Process

Visualizing Repository Data with Graphs

Repository graphs

Repository graphs help you view and analyze data for your repository.

- Pulse
- Contributors
- Traffic
- Commits
- Code frequency
- Network

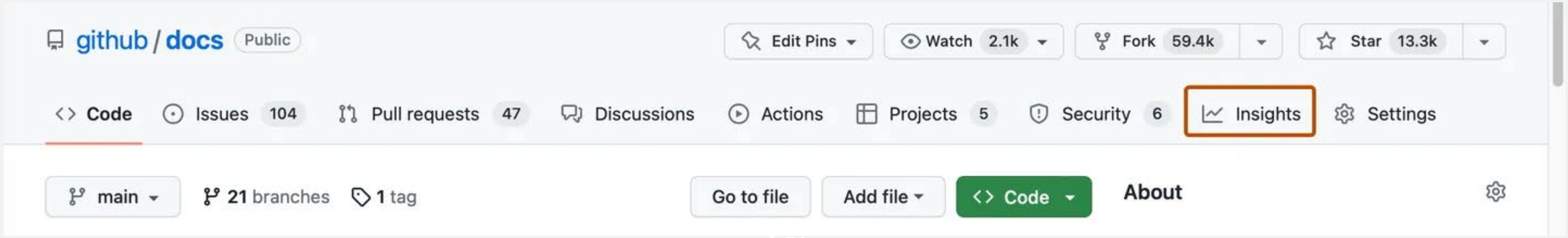
Visualizing Repository Data with Graphs

Pulse

You can use Pulse to see an overview of a repository's
pull request, issue, and commit activity.

1. On GitHub.com, navigate to the main page of the repository.

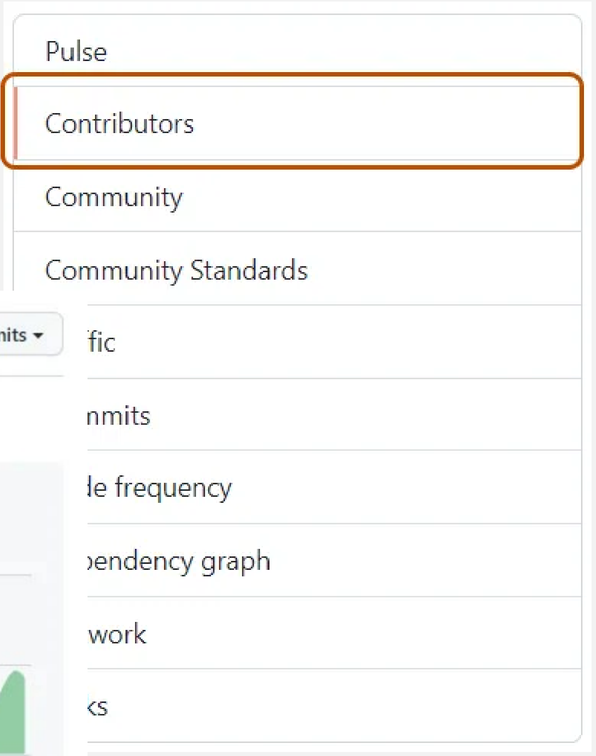
2. Under your repository name, click Insights.

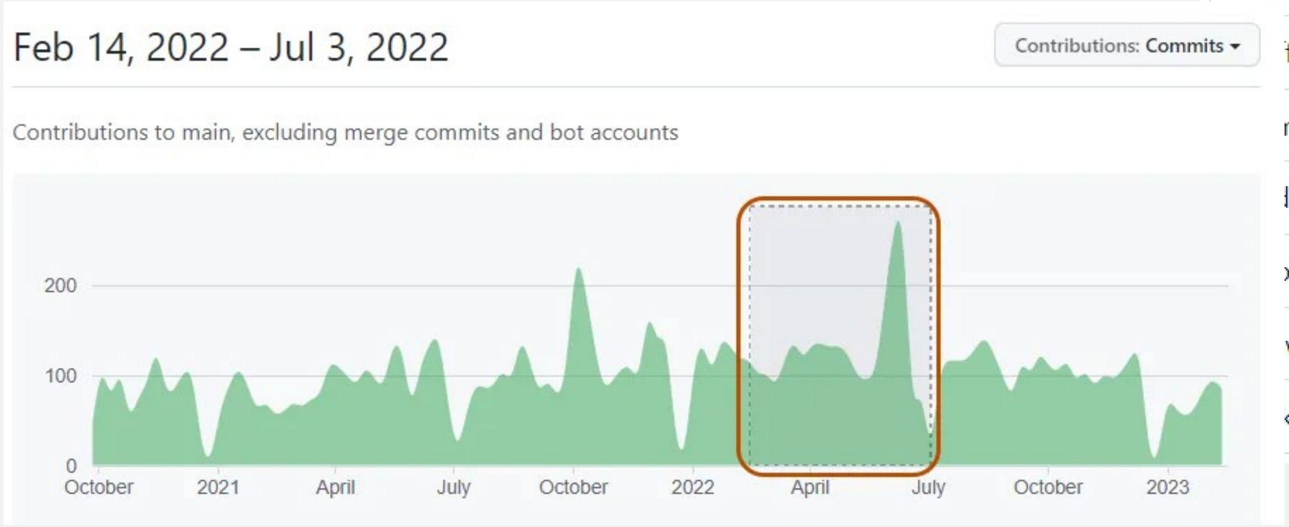


Visualizing Repository Data with Graphs

Contributors Graph

You can see who contributed commits to a repository and its dependencies.



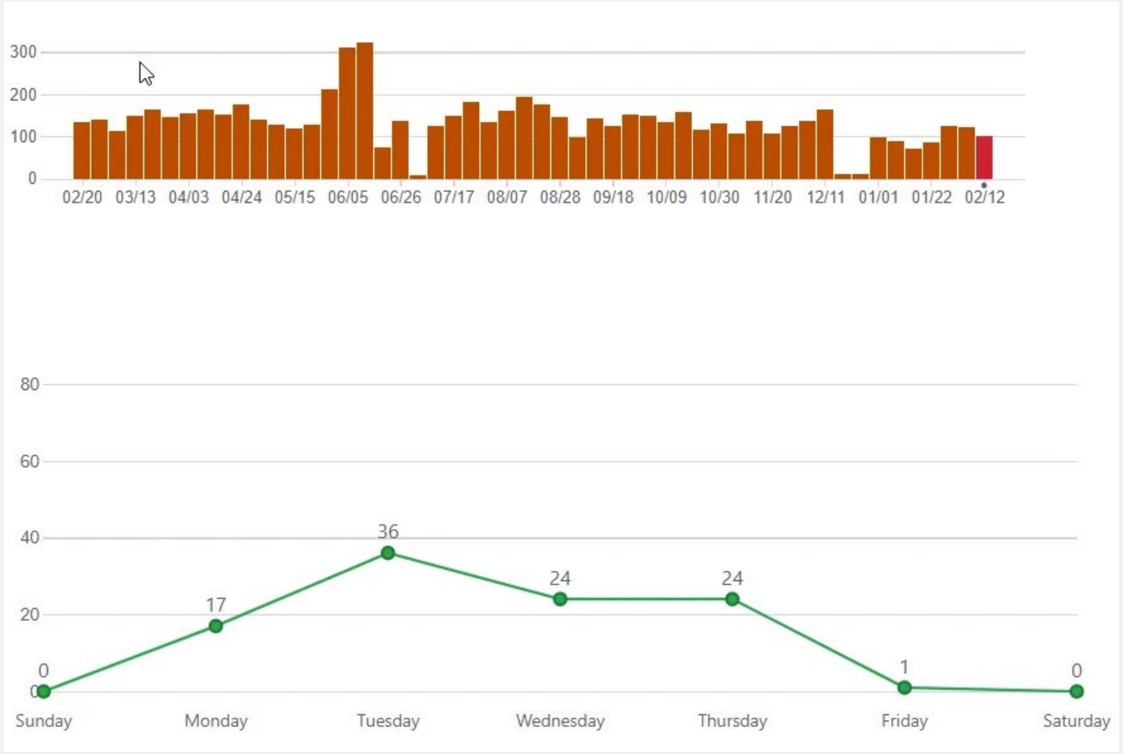

Feb 14, 2022 – Jul 3, 2022

Contributions to main, excluding merge commits and bot accounts

Visualizing Repository Data with Graphs

Commit Activity Graph

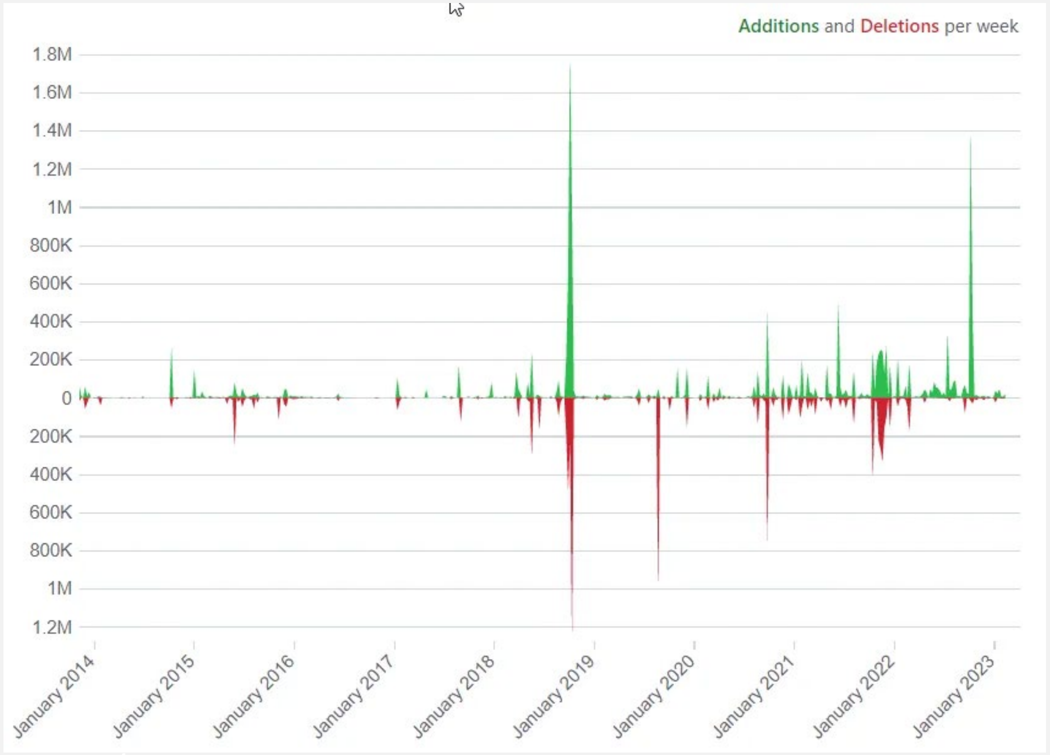
Analyzing changes to a repository's content. You can see the changes to the content of a repository by analyzing the repository's commits, commit frequency, and content additions and deletions.



Visualizing Repository Data with Graphs

Code frequency

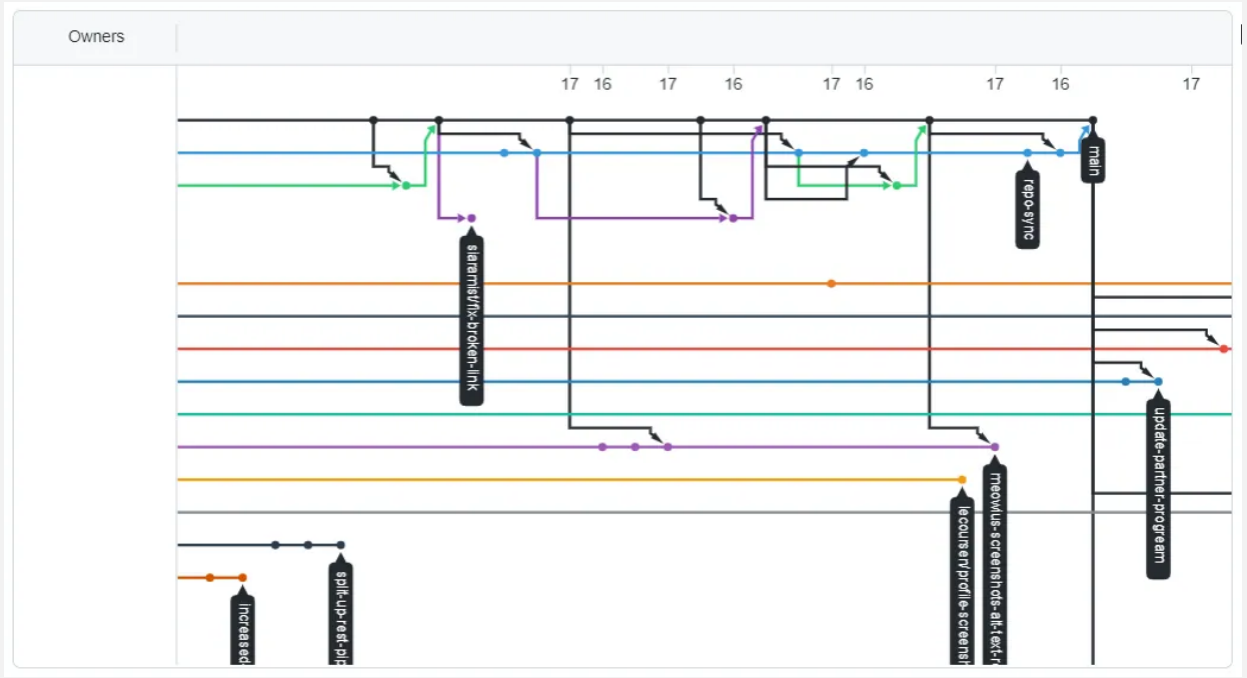
Visualizing additions and deletion to content in a repository



Visualizing Repository Data with Graphs

Network

Understanding connections between repositories



Using insights for project management

GitHub provides a variety of tools and insights that help project managers and developers monitor the progress, health, and contributions of their projects. These insights are critical for effective project management, allowing you to make data-driven decisions and maintain a well-organized and productive workflow.

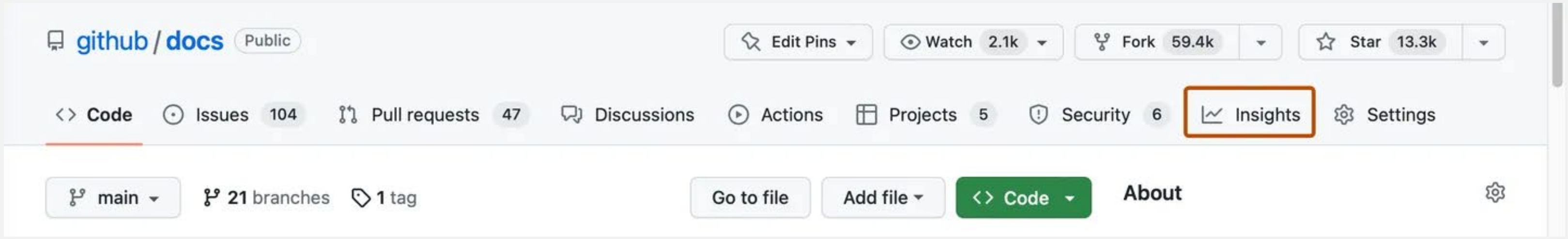
Using insights for project management

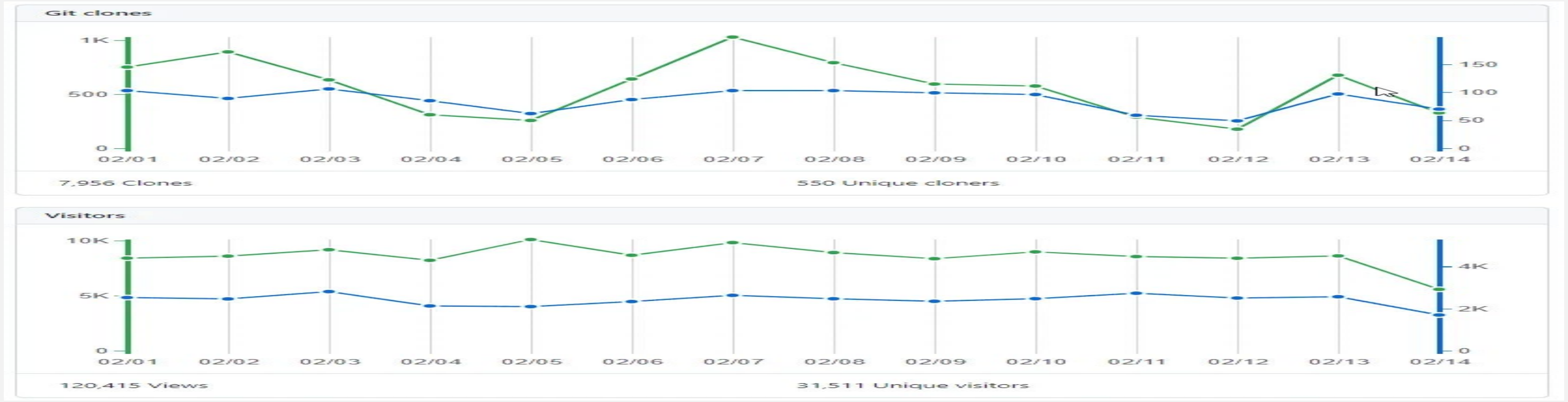
Issue and Pull Request Insights on GitHub

Key Features of Issue and Pull Request Insights

- Tracking Progress
- Identifying Bottlenecks
- Prioritizing Work
- Collaborative Feedback
- Activity Overview

Viewing traffic to a repository





7,956 Clones

550 Unique cloners

120,415 Views

31,511 Unique visitors

Using insights for project management

Issue and Pull Request Insights on GitHub

Key Features of Issue and Pull Request Insights

- Tracking Progress
- Identifying Bottlenecks
- Prioritizing Work
- Collaborative Feedback
- Activity Overview

6. Search and Integration

Search and Integration

GitHub provides powerful search tools that help you find files, pull requests (PRs), issues, and code across repositories. By mastering these search techniques, you can quickly locate the information you need and enhance your efficiency.

Basic Search

Search Bar:

- The search bar at the top of GitHub pages allows you to perform basic searches.
- You can search for repositories, users, issues, code, and more.

Search and Integration

Advanced Search

GitHub's advanced search options allow you to refine your searches using specific qualifiers.

Search Qualifiers:

Repositories: repo:<user>/<repo>: Search within a specific repository.

Search and Integration

Advanced Search

Search Qualifiers:

Code:

- language:<language>: Search for code in a specific language.
- path:<path>: Search within a specific directory.
- filename:<filename>: Search for a specific file.

Search and Integration

Advanced Search

Search Qualifiers:

Issues and PRs:

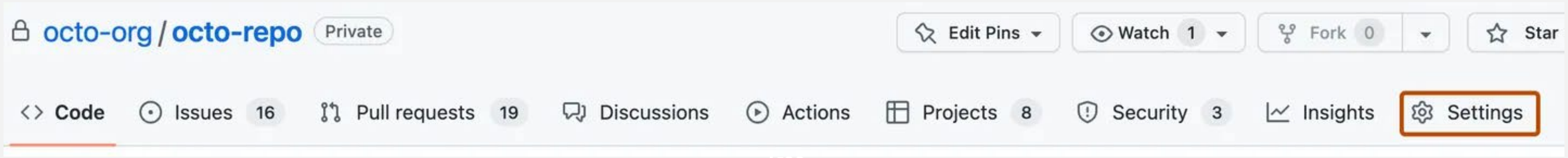
- is:issue or is:pr: Search for issues or pull requests.
- is:open or is:closed: Search for open or closed items.
- author:<username>: Search for issues or PRs created by a specific user.
- label:<label>: Search for issues or PRs with a specific label.
- assignee:<username>: Search for issues or PRs assigned to a specific user.

Integrations and Automation

Setting up webhooks and third-party integrations

Webhooks:

Webhooks allow you to receive real-time notifications about events in your GitHub repository. They can trigger actions on external services based on repository events such as pushes, pull requests, and issues.



Integrations and Automation

Setting up webhooks and third-party integrations

Example Use Cases:

•Continuous Integration: Trigger CI/CD pipelines on platforms like Jenkins, Travis CI, or CircleCI.

•Notifications: Send notifications to Slack or email when issues or pull requests are created or updated.

•Automation: Automatically deploy code to production or update documentation.

Integrations and Automation

Third-Party Integrations

GitHub integrates with numerous third-party services to enhance your workflow.

Popular Integrations:

- Slack
- Jira
- Travis CI
- Docker Hub

Integrations and Automation

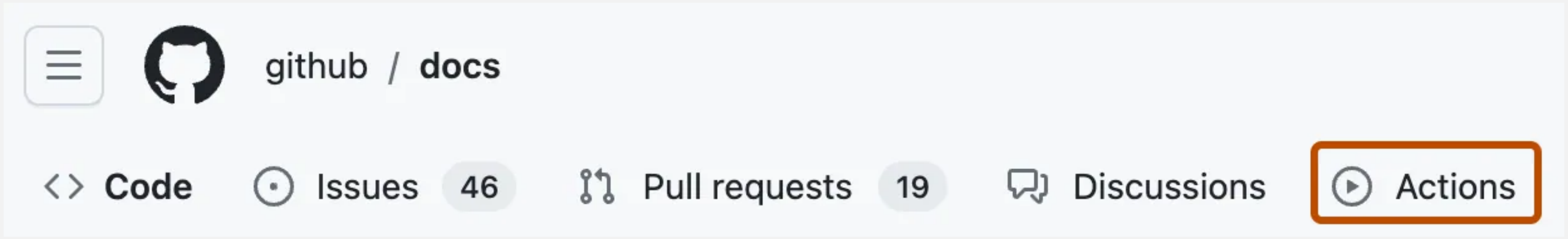
Third-Party Integrations

Example Use Cases:

- Automated Testing: Use Travis CI to run tests on every push to your repository.
- Collaboration: Use Slack integration to keep your team informed about repository activity.
- Issue Tracking: Use Jira integration to link code changes with project management tasks.

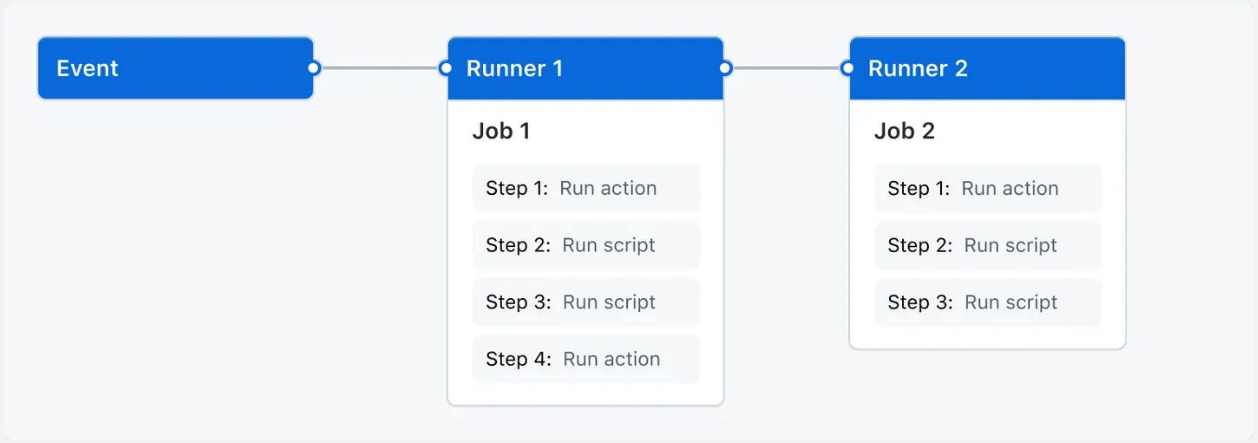
Integrations and Automation

Integrating GitHub with third-party services and automating workflows using GitHub Actions can greatly enhance your development process. These tools allow you to automate tasks, streamline collaboration, and ensure consistent project management.



Integrations and Automation

Automating workflows with GitHub Actions



GitHub Actions is a continuous integration and continuous delivery (CI/CD) platform

Integrations and Automation

Automating workflows with GitHub Actions

Workflows

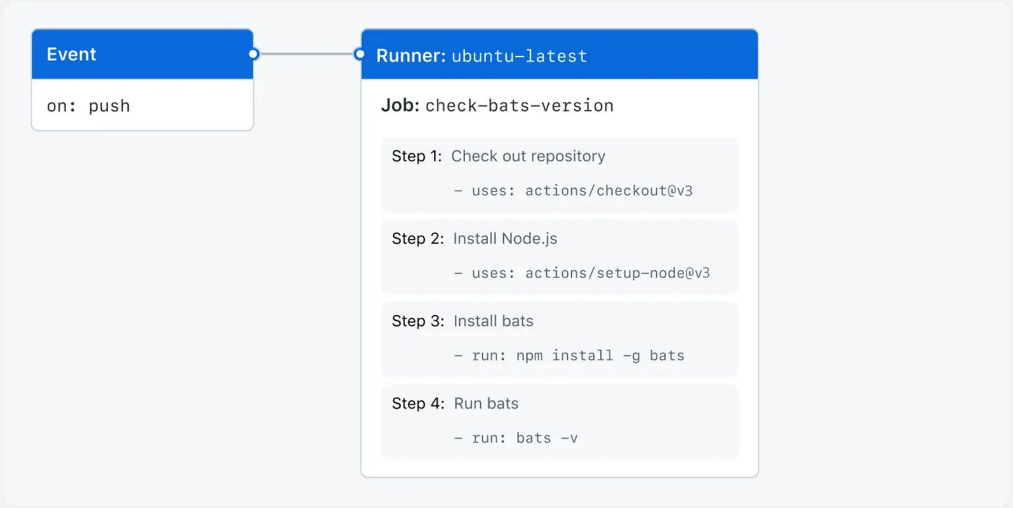
A workflow is a configurable automated process that will run one or more jobs. Workflows are defined by a YAML file checked in to your repository and will run when triggered by an event in your repository, or they can be triggered manually, or at a defined schedule.

Integrations and Automation

Automating workflows with GitHub Actions

Events

An event is a specific activity in a repository that triggers a workflow run. For example, an activity can originate from GitHub when someone creates a pull request, opens an issue, or pushes a commit to a repository.



```
on: push
- uses: actions/checkout@v3
- uses: actions/setup-node@v3
- run: npm install -g bats
- run: bats -v
```

Integrations and Automation

Automating workflows with GitHub Actions

Jobs

A job is a set of steps in a workflow that is executed on the same runner. Each step is either a shell script that will be executed, or an action that will be run. Steps are executed in order and are dependent on each other. Since each step is executed on the same runner, you can share data from one step to another. For example, you can have a step that builds your application followed by a step that tests the application that was built.

Integrations and Automation

Automating workflows with GitHub Actions

Actions

An action is a custom application for the GitHub Actions platform that performs a complex but frequently repeated task. Use an action to help reduce the amount of repetitive code that you write in your workflow files. An action can pull your git repository from GitHub, set up the correct toolchain for your build environment, or set up the authentication to your cloud provider.

Integrations and Automation

Automating workflows with GitHub Actions

Runners

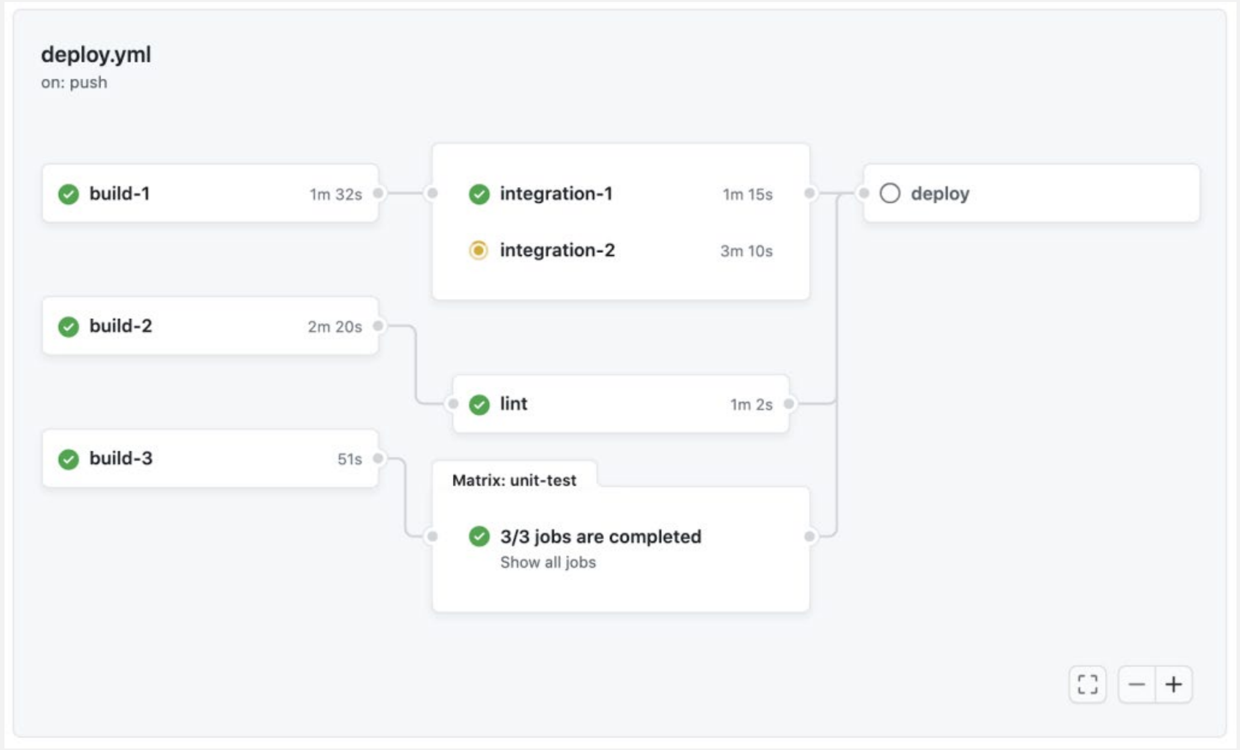
A runner is a server that runs your workflows when they're triggered. Each runner can run a single job at a time. GitHub provides Ubuntu Linux, Microsoft Windows, and macOS runners to run your workflows; each workflow run executes in a fresh, newly-provisioned virtual machine.

Complex Workflows

- GitHub Actions is designed to bring platform-native automation and CI/CD capabilities directly into the GitHub flow to simplify the developer experience.

- It can also be used to build out more advanced, custom workflows for anything from triggering an alarm to orchestrating complex security test automations.

Track your workflows with workflow visualization



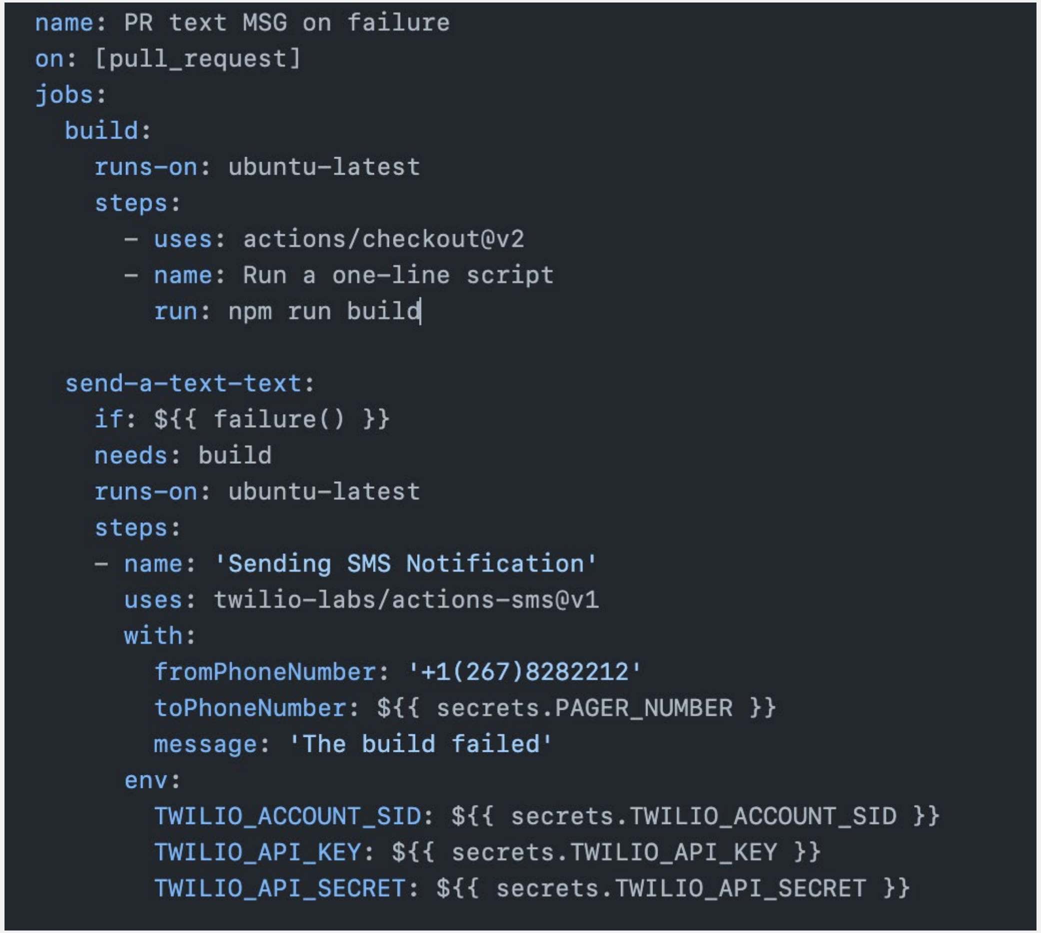
Build breakpoints in a workflow with dependencies

- GitHub Actions runs multiple commands at the same time by default—but you can leverage the needs keyword to create dependencies between jobs.
- That means that if something like a test fails (or any job for that matter) dependent jobs won’t execute.

A conditional can make all the difference

- GitHub Actions supports conditionals, which use the if keyword to determine if a step should run in a given workflow.
- You can use this to build upon dependencies so that if a dependent job fails the workflow can continue running.
- These can use certain built-in functions for data operations.

A conditional can make all the difference



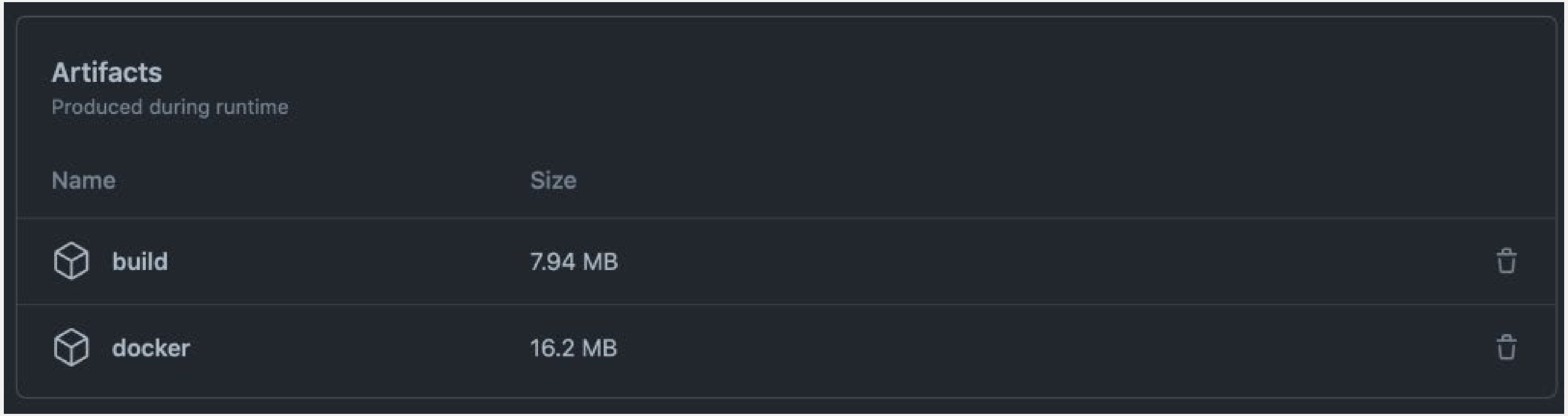
```yaml
name: PR text MSG on failure
on: [pull_request]
jobs:
  build:
    runs-on: ubuntu-latest
    steps:
      - uses: actions/checkout@v2
      - name: Run a one-line script
        run: npm run build

  send-a-text-text:
    if: ${{ failure() }}
    needs: build
    runs-on: ubuntu-latest
    steps:
    - name: 'Sending SMS Notification'
      uses: twilio-labs/actions-sms@v1
      with:
        fromPhoneNumber: '+1(267)8282212'
        toPhoneNumber: ${{ secrets.PAGER_NUMBER }}
        message: 'The build failed'
      env:
        TWILIO_ACCOUNT_SID: ${{ secrets.TWILIO_ACCOUNT_SID }}
        TWILIO_API_KEY: ${{ secrets.TWILIO_API_KEY }}
        TWILIO_API_SECRET: ${{ secrets.TWILIO_API_SECRET }}
```

Creating workflows with sensitive data? Try storing secrets

- Secrets on GitHub are used to securely store sensitive data such as passwords, tokens, and certificates, among other things—and they can be directly referenced in workflows, too.

- This means you can create and share workflows with collaborators using secrets for secure values without hardcoding those values directly in YAML files.

Share data between jobs to create more advanced automations



Artifacts

Produced during runtime

Name	Size
build	7.94 MB
docker	16.2 MB

Collaboration and Team Management

Introduction to Collaboration and Team Management

Setting Up and Managing Organizations and Teams

Organizations on GitHub provide a centralized way to manage multiple repositories and users.

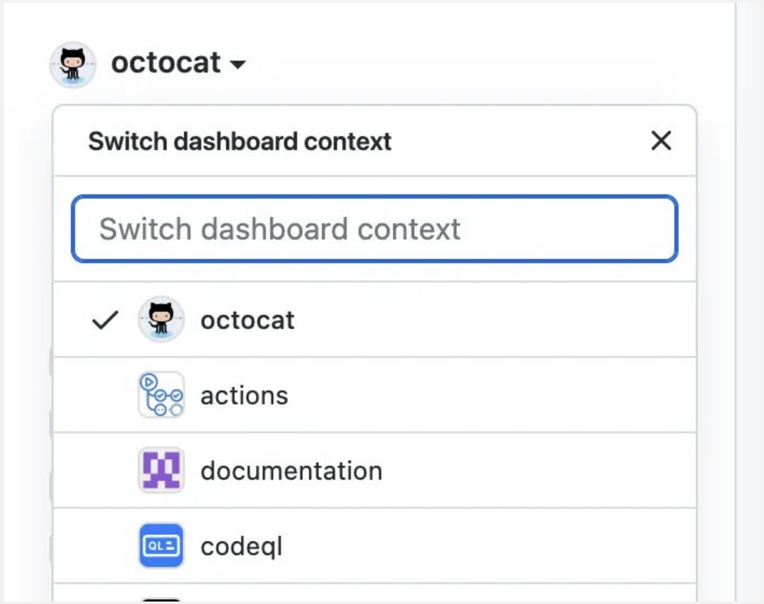
Benefits:

• Centralized management of repositories and team members.
• Improved collaboration through shared access and permissions.

Setting Up and Managing Organizations and Teams

About your organization dashboard

As an organization member, you can visit your organization's dashboard throughout the day to stay updated on recent activity and keep track of issues and pull requests you're working on or following in the organization



Setting Up and Managing Organizations and Teams

Best practices for organizations

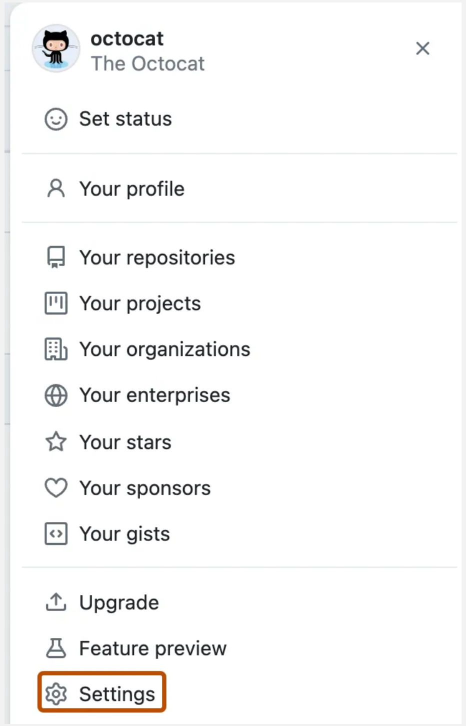
- Assign multiple owners
    - If an organization only has one owner, the organization's projects can become inaccessible if the owner is unreachable.
- Use teams to facilitate collaboration in your organization.

Setting Up and Managing Organizations and Teams

Creating a new organization

Create an organization to apply fine-grained access permissions to repositories.

In the upper-right corner of any page, click your profile photo, then click Settings.



Setting Up and Managing Organizations and Teams

Accessing your organization's settings

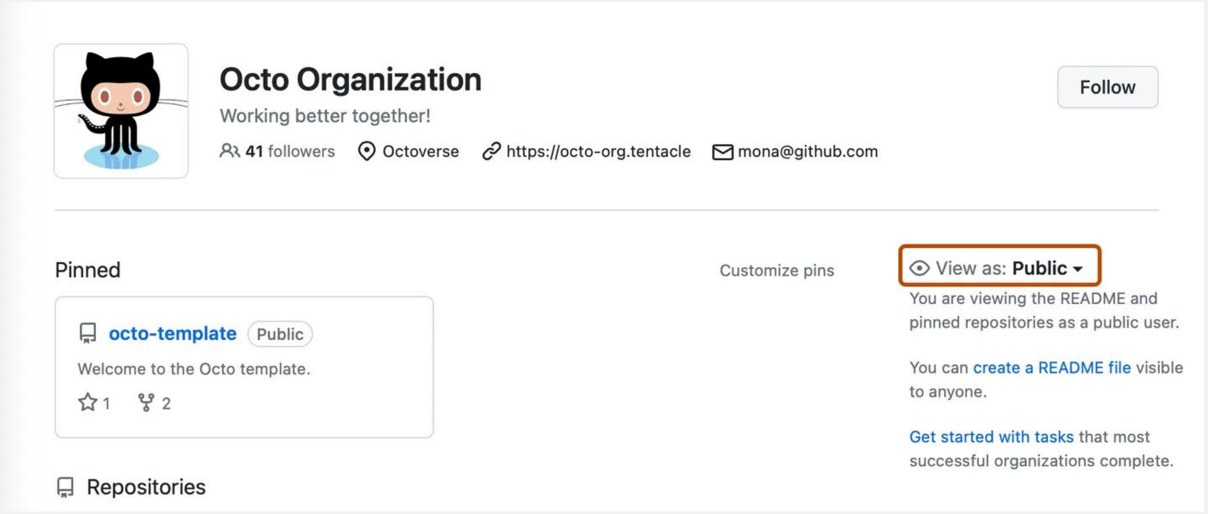
The organization account settings page provides several ways to manage the account, such as billing, team membership, and repository settings.

- In the upper-right corner of GitHub, select your profile photo, then click Your organizations.
- Next to the organization, click Settings.

Setting Up and Managing Organizations and Teams

Customizing your organization's profile

You can share information about your organization by customizing your organization's profile.



Octo Organization

Working better together!

41 followers · Octoverse · https://octo-org.tentacle · mona@github.com

Follow

Pinned

Customize pins

View as: Public

You are viewing the README and pinned repositories as a public user.

You can create a README file visible to anyone.

Get started with tasks that most successful organizations complete.

octo-template Public

Welcome to the Octo template.

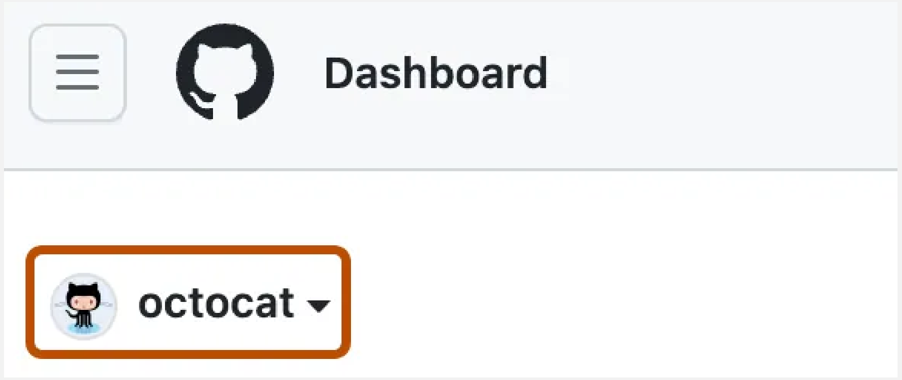
1 · 2

Repositories

Setting Up and Managing Organizations and Teams

About your organization’s news feed

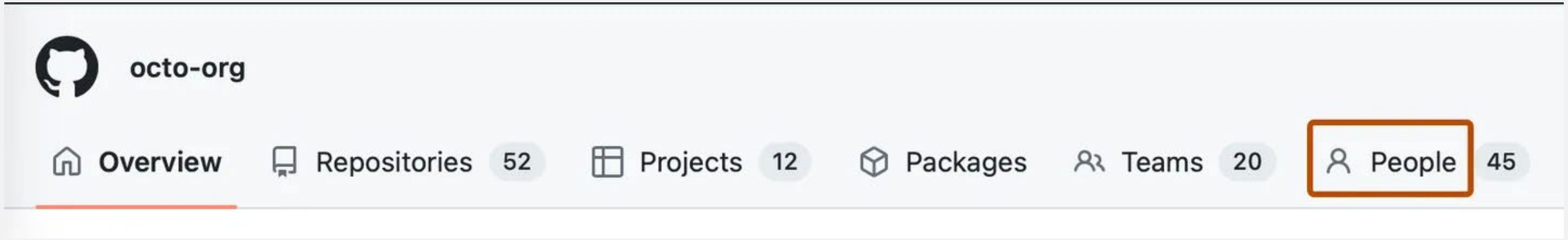
You can use your organization's news feed to keep up with recent activity on repositories owned by that organization



Setting Up and Managing Organizations and Teams

Inviting users to join your organization

You can invite anyone to become a member of your organization using their username or email address for GitHub.com.



Setting Up and Managing Organizations and Teams

Removing a member from your organization

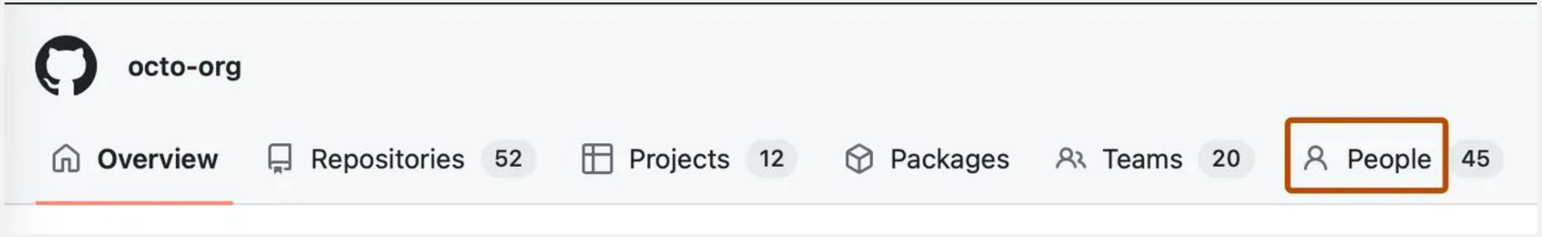
If members of your organization no longer require access to any repositories owned by the organization, you can remove them from the organization.

Organization owners can remove members from an organization.

Setting Up and Managing Organizations and Teams

Reinstating a former member of your organization

You can invite former organization members to rejoin your organization, and choose whether to restore the person's former role, access permissions, forks, and settings



Setting Up and Managing Organizations and Teams

Can I create accounts for people in my organization?

While you can add users to an organization you've created, you can't create personal accounts on behalf of another person.

Setting Up and Managing Organizations and Teams

Roles in an organization

Organization owners can assign roles to individuals and teams giving them different sets of permissions in the organization.

- Organization owners
- Organization members
- Organization moderators
- Billing managers
- Security managers
- GitHub App managers
- Outside collaborators

Setting Up and Managing Organizations and Teams

Organization owners

Organization owners have complete administrative access to your organization. This role should be limited, but to no less than two people, in your organization.

Organization members

The default, non-administrative role for people in an organization is the organization member. By default, organization members have a number of permissions, including the ability to create repositories and projects.

Setting Up and Managing Organizations and Teams

Organization moderators

Moderators are organization members who, in addition to their permissions as members, are allowed to block and unblock non-member contributors, set interaction limits, and hide comments in public repositories owned by the organization.

Billing managers

Billing managers are users who can manage the billing settings for your organization, such as payment information. This is a useful option if members of your organization don't usually have access to billing resources.

Setting Up and Managing Organizations and Teams

Security managers

Security manager is an organization-level role that organization owners can assign to any team in an organization. When applied, it gives every member of the team permissions to view security alerts and manage settings for code security across your organization, as well as read permissions for all repositories in the organization

Setting Up and Managing Organizations and Teams

GitHub App managers

By default, only organization owners can manage the settings of GitHub App registrations owned by an organization. To allow additional users to manage GitHub App registrations owned by an organization, an owner can grant them GitHub App manager permissions.

Outside collaborators
To keep your organization's data secure while allowing access to repositories, you can add outside collaborators. An outside collaborator is a person who has access to one or more organization repositories but is not explicitly a member of the organization, such as a consultant or temporary employee.

Setting Up and Managing Organizations and Teams

Managing security managers in your organization

Security manager is an organization-level role that organization owners can assign to any team in an organization. When applied, it gives every member of the team permissions to view security alerts and manage settings for code security across your organization, as well as read permissions for all repositories in the organization.

Setting Up and Managing Organizations and Teams

Managing moderators in your organization

Sometimes it's necessary to block a contributor, or to set up interaction limits for your organization, or for individual repositories. As an organization owner, you can perform these tasks, but you may want to delegate these tasks to other members of your organization. You can do this by assigning an organization member, or a team, to the moderator role.

Setting Up and Managing Organizations and Teams

Repository roles for an organization

You can customize access to each repository in your organization by assigning granular roles, giving people access to the features and tasks they need.

From least access to most access, the roles for an organization repository are:

- **Read**
- **Triage**
- **Write**
- **Maintain**
- **Admin**

Setting Up and Managing Organizations and Teams

Setting base permissions for an organization

You can set base permissions that apply to all members of an organization when accessing any of the organization's repositories. Base permissions do not apply to outside collaborators.

By default, members of an organization will have Read permissions to the organization's repositories.

Setting Up and Managing Organizations and Teams

Viewing people with access to your repository

You can view a list of people with access to a repository within an organization.

Who can use this feature?
Organization owners can view people with access to a repository.

You can see a combined overview of teams and people with access to your repository in your repository settings.

Setting Up and Managing Organizations and Teams

Managing an individual's access to an organization repository

When you remove a collaborator from a repository in your organization, the collaborator loses read and write access to the repository. If the repository is private and the collaborator has forked the repository, then their fork is also deleted, but the collaborator will still retain any local clones of your repository.

Setting Up and Managing Organizations and Teams

Managing team access to an organization repository

People with admin access to a repository can manage team access to the repository. Team maintainers can remove a team's access to a repository if the team has direct access to it. If the team's access to the repository is inherited from a parent team, maintainers can choose to reset the current permission to match the parent team's permission.

Setting Up and Managing Organizations and Teams

Adding outside collaborators to repositories in your organization

Who can use this feature?
People with admin access to a repository can add an outside collaborator to the repository.

An outside collaborator is a person who is not a member of your organization, but has access to one or more of your organization's repositories. You can choose the level of access to grant for each outside collaborator. When you add an outside collaborator to a repository, you'll also need to add them to any forks of the repository you'd like them to access.

Setting Up and Managing Organizations and Teams

Purpose of Teams in GitHub Organizations

•**Structured Collaboration**: Teams provide a structured way to manage groups of users, making it easier to assign tasks, review code, and manage projects.

•**Access Control**: Teams enable precise control over who has access to which repositories and what level of access they have (read, write, admin).

•**Streamlined Communication**: Teams facilitate better communication and coordination among members working on specific projects or areas of a project.

Collaborating with Issues and Pull Requests

Collaborating with issues and pull requests (PRs) on GitHub is essential for managing software development projects.

Issues: Used for tracking tasks, enhancements, bugs, and feature requests.

Pull Requests: Used for proposing code changes, conducting code reviews, and merging changes into the main codebase.

Collaborating with Issues and Pull Requests

Using Issues for Project Tracking
Purpose of Issues

Issues on GitHub help you manage and organize work by providing a way to track tasks, enhancements, bugs, and feature requests. They are essential for project management and collaboration.

Benefits:

- Centralized tracking of tasks and bugs.
- Facilitates discussion and documentation of issues.
- Allows for easy prioritization and assignment of tasks.

Collaborating with Issues and Pull Requests

Using Pull Requests for Code Review
Purpose of Pull Requests

Pull requests (PRs) allow you to propose changes to the codebase, conduct code reviews, and merge changes. They are crucial for maintaining code quality and enabling collaborative development.

Benefits:

- Facilitates code reviews and feedback.
- Tracks proposed changes and discussions.
- Integrates with continuous integration (CI) systems for automated testing.

Collaborating with Issues and Pull Requests

Best Practices for Using Issues and Pull Requests

- Detailed Descriptions: Provide clear and detailed descriptions for issues and PRs to ensure everyone understands the context.
- Consistent Labels: Use a consistent labeling system to categorize and prioritize issues and PRs.
- Regular Updates: Keep issues and PRs updated with the latest information and progress.
- Automated Checks: Integrate CI tools to automatically run tests and checks on PRs.
- Collaborative Reviews: Encourage team members to participate in code reviews and discussions.

GitHub Security Features

Understanding GitHub Security Basics

Defining Vulnerabilities:

- A vulnerability is a weakness in the software that can be exploited to cause harm.
- Vulnerabilities can be due to coding errors, misconfigurations, or insufficient security measures.

Understanding GitHub Security Basics

Defining Vulnerabilities:

1. Coding Errors
2. Configuration Issues
3. Dependency Vulnerabilities
4. Insecure Practices
5. Design Flaws

Common Sources of Vulnerabilities

- Human Error: Mistakes made by developers, such as coding errors or misconfigurations.
- Complexity: Complex systems are harder to secure due to the increased number of components and interactions.
- Third-Party Code: Dependencies and third-party libraries may introduce vulnerabilities.
- Evolving Threats: New vulnerabilities and attack vectors emerge regularly.

Understanding GitHub Security Basics

Mitigating Vulnerabilities

- Regular Updates: Keep your dependencies up to date to ensure you have the latest security patches.
- Code Reviews: Conduct regular code reviews to identify and fix potential vulnerabilities.
- Static Analysis: Use static analysis tools to detect vulnerabilities in your code.
- Security Policies: Implement and enforce security policies to guide development practices.
- Training: Educate developers about common security threats and secure coding practices.

Understanding GitHub Security Basics

Vulnerable Dependencies

Dependencies are external libraries, frameworks, or modules that your project relies on to function. They can significantly speed up development by providing pre-built solutions for common tasks. However, dependencies can also introduce vulnerabilities into your project if they contain security flaws.

Understanding GitHub Security Basics

Identifying Vulnerabilities on GitHub

GitHub provides several tools to help identify and manage vulnerabilities in your projects:

Dependabot Alerts
- Function: Scans your dependencies and alerts you to known vulnerabilities.

Code Scanning
- Function: Analyzes your code for potential security issues using tools like CodeQL.

Secret Scanning
- Function: Detects sensitive information such as API keys and passwords in your repositories.

Understanding GitHub Security Basics

Importance of a Security Policy

Definition and Purpose of a Security Policy

A security policy is a formal document that defines how an organization protects its information assets and manages security risks. It provides a framework for identifying, managing, and mitigating security threats and vulnerabilities.

Understanding GitHub Security Basics

Importance of Implementing a Security Policy

•Preventing Security Breaches
•Ensuring Compliance
•Risk Management
•Protecting Reputation
•Promoting Security Awareness

Advanced Security Tools and Practices

Secret Scanning

Secret Scanning is a security feature that helps detect and prevent the accidental exposure of sensitive information such as API keys, passwords, and tokens in your codebase. Exposing these secrets can lead to unauthorized access to services and data breaches.

- Prevent Unauthorized Access
- Compliance
- Maintain Trust

Advanced Security Tools and Practices

Code Scanning and Automation

Code Scanning is the process of automatically analyzing your codebase to identify potential security vulnerabilities and code quality issues.

Need for Code Scanning:

•Early Detection: Identify and fix security vulnerabilities and bugs early in the development process.
•Continuous Monitoring: Ensure ongoing security by continuously scanning code changes.
•Compliance: Meet regulatory requirements for secure coding practices.

Understanding GitHub Security Basics

Dependency Management

Dependency Management involves tracking and updating the external libraries and packages that your project depends on. The dependency graph is a visualization of these dependencies and their relationships.

Dependency Graph:

- Visualization
- Transitive Dependencies
- Security:

Implementing and Managing Security Measures

Security Vulnerability Alerts

GitHub provides automated security alerts to help repository owners manage vulnerabilities in their dependencies. These alerts are part of GitHub's security features and are designed to help you quickly identify and mitigate potential security risks.

Implementing and Managing Security Measures

Security Advisories

Security Advisories are detailed reports about security vulnerabilities found in a software project. They provide information about the nature of the vulnerability, its impact, and steps to mitigate or fix the issue.

- Communication
- Transparency
- Guidance

Implementing and Managing Security Measures

Creating and Implementing a Security Response Plan

A Security Response Plan is a documented strategy outlining the steps to take in response to a security incident, particularly data breaches. This plan ensures a structured and effective approach to managing and mitigating security incidents.

Access and Data Protection

Steps to Create a Security Response Plan:

1. Define Objectives
2. Assemble a Response Team
3. Identify Critical Assets
4. Introduction:
5. Response Team:
6. Detection:
7. Containment:
8. Recovery:
9. Communication:
10. Post-Incident:

Access and Data Protection

Using SSH and Deploy Keys

SSH keys and Deploy keys are essential tools for securely accessing and interacting with GitHub repositories. They help ensure that data exchanges between your local machine and GitHub are encrypted and secure.

Deploy keys are SSH keys that are added directly to a repository rather than a user account.
They allow read or read/write access to a single repository and are typically used for automated deployments or continuous integration (CI) systems.