# GitHub

## GitHub

### Definition:

GitHub is a platform for version control and collaboration. It is used by millions of developers around the world to store, track, and manage code.

### Features:

1. Code Hosting: Store your code in the cloud, allowing you to access and manage it from anywhere.
2. Version Control: Track changes to your code over time, enabling you to revert to previous versions if needed.
3. Collaboration: Work together with others on projects by sharing code, reviewing changes, and discussing ideas.
4. Issue Tracking: Manage tasks, bugs, and feature requests within your projects, providing a centralized hub for communication.

### Advantages:

Facilitates collaboration with features like pull requests, issue tracking, and code reviews. It integrates with various tools for CI/CD, offers hosting for documentation, and supports open-source contributions, all while providing a centralized platform for managing Git repositories.

### Disadvantages:

Limited features on free plans, with more advanced options reserved for paid tiers. Public repositories expose code unless a private plan is used, and there can be a learning curve for beginners unfamiliar with Git or GitHub's interface.

### Working:

GitHub hosts Git repositories in the cloud, enabling collaboration through pull requests, issue tracking, and code reviews. Developers push their code to remote repositories on GitHub, where teams can manage projects, track changes, and collaborate seamlessly. GitHub integrates with CI/CD tools, automating testing and deployment directly from the repository.

## GitHub Account

### Definition:

A GitHub Account is a personal or organizational profile on GitHub that allows users to create, manage, and collaborate on repositories, enabling them to contribute to projects, follow other users, and engage with the GitHub community.

### Features:

Provides access to create and manage repositories, collaborate with others through pull requests and issues, customize profiles, access GitHub Actions for automation, and integrate with third-party tools and services.

### Advantages:

Centralizes all your projects and contributions, facilitates collaboration with other developers, and provides a professional online presence. It also allows seamless access to GitHub's vast ecosystem of tools, integrations, and community resources.

### Disadvantages:

Public repositories may expose code if not carefully managed, and the advanced features are only available on paid plans. The platform's complexity might be overwhelming for beginners, and account security must be diligently maintained to prevent unauthorized access.

### Types:

1. **Personal Account:** Designed for individual developers, personal accounts offer features like private repositories, collaboration with other users, and project management tools. Personal Accounts can be classified as:
2. **Free Account:** Provides public repositories, unlimited collaborators, issue tracking, and basic project management features.
3. **Pro Account:** Offers features like private repositories, advanced code review tools, and code scanning for security vulnerabilities.
4. **Organization Account:** Ideal for teams and businesses, organization accounts provide advanced collaboration features, team management capabilities, and enterprise-grade security. Organization Accounts can be classified as:
5. **Free for Organizations:** Provides features like private repositories, unlimited collaborators, issue tracking, and team management capabilities.
6. **GitHub Teams:** Offers more advanced features like team-specific permissions, code ownership, and advanced analytics for team performance.
7. **Enterprise Account:** Catering to larger organizations, Enterprise accounts offer robust features like on-premises deployment, advanced security controls, and enterprise-level support. Enterprise Accounts can be classified as:
8. **GitHub Enterprise Server:** Allows organizations to host their own GitHub Enterprise instance on their own infrastructure for enhanced security and control.
9. **GitHub Enterprise Cloud:** Provides a fully managed, cloud-based solution for GitHub Enterprise, offering scalability, reliability, and ease of deployment.

### Working:

A GitHub Account allows users to sign in, create repositories, and collaborate with others. It provides a dashboard to manage projects, notifications, and contributions. Users can configure settings, manage access permissions, and integrate with other services to streamline development workflows.

## Creating a GitHub Account

1. **Sign Up:** Visit the GitHub website and click on the "Sign Up" button to create a new account.
2. **Choose a Username:** Select a unique username that represents you or your brand.
3. **Verify Email:** Confirm your email address to activate your account and get started.

## GitHub Authentication

### Definition:

GitHub Authentication is the process of verifying a user’s identity to grant access to their GitHub account and repositories, ensuring that only authorized users can interact with the platform.

### Features:

Includes secure login options like username/password, OAuth, SSH keys, and multi-factor authentication (MFA). It supports personal access tokens (PATs) for API access and OAuth tokens for third-party app integrations.

### Advantages:

Enhances security by ensuring only authorized users can access repositories and account settings. Offers flexibility with multiple authentication methods and supports secure API and third-party integrations.

### Disadvantages:

Managing multiple authentication methods can be complex and may require extra setup, particularly for MFA. Tokens and keys must be securely stored, as their compromise can lead to unauthorized access.

### Types:

1. **Username and Password:** Standard login method.
2. **SSH Keys:** Secure, passwordless authentication for Git operations.
3. **OAuth:** Token-based access for third-party applications.
4. **Personal Access Tokens (PATs):** Used for API access and replacing passwords for Git operations.
5. **Multi-Factor Authentication (MFA):** Adds an extra layer of security by requiring a second form of verification.

### Working:

Users authenticate by entering their credentials or using an authorized token/key. GitHub verifies the credentials or token against its database, and if valid, grants access to the user’s account and repositories. MFA, if enabled, requires an additional verification step, such as entering a code from an authentication app. For Git operations, SSH keys or PATs authenticate the user, allowing secure interaction with the repository.

## Authenticate GitHub Account using Username and Password

This method relies on a traditional username and password combination. It is the most common method but considered less secure than others.

Step 1: Enter a unique Username.

Step 2: Enter Email Address.

Step 3: Create a Password.

Step 4: Verify Account

Step 5: Choose a Plan

Step 6: Customize Experience

Step 7: Confirm Email Address

## Authenticate GitHub Account using SSH Keys

SSH keys provide a more secure alternative to password-based authentication. You generate a pair of keys, one public and one private, and store the public key on GitHub.

Step 1: Generate Keys: Create a key pair using SSH commands.

Step 2: Add Public Key: Copy the public key and add it to your GitHub account settings.

Step 3: Authenticate: Use the private key to authenticate with GitHub.

## Authenticate GitHub Account using OAuth

OAuth delegates authentication to a third-party service. You log in through a trusted service, granting GitHub temporary access to your account. It is used for seamless integration with other applications.

Step 1: Request access to GitHub.

Step 2: GitHub redirects to a third-party service.

Step 3: Login to the third-party service.

Step 4: GitHub authorizes access and returns you to the application.

## Authenticate GitHub Account using Personal Access Tokens (PATs)

PATs are unique tokens that provide limited access to your GitHub account. You can use them to automate tasks or access certain features without needing to provide your username or password.

Step 1: Create Token: Generate a PAT with specific permissions.

Step 2: Use Token: Utilize the token in API calls or GitHub applications.

Step 3: Revoke Token: Remove access by revoking the token when no longer needed.

## Authenticate GitHub Account using Multi-Factor Authentication (MFA)

MFA adds an extra layer of security to your account by requiring you to provide a second factor of

authentication, such as a code from your phone or a security key, in addition to your username and password. When you log in, you need a secondary factor, like a code from your phone. This makes it much harder for unauthorized users to access your account.

Step 1: Enter username and password.

Step 2: GitHub requests an MFA code.

Step 3: Enter the code from your phone or authenticator app.

Step 4: GitHub verifies the code and grants access.

## GitHub Security

### Definition:

GitHub Security encompasses the tools, features, and practices provided by GitHub to protect code, repositories, and accounts from vulnerabilities, unauthorized access, and malicious activity.

### Features:

Includes dependency scanning, secret scanning, security alerts, code scanning, branch protection rules, two-factor authentication (2FA), and security advisories. It also supports role-based access control (RBAC) and secure Git operations through SSH and HTTPS.

### Advantages:

Provides comprehensive protection for code and repositories, helping to identify and mitigate vulnerabilities early. It ensures secure access to repositories and enhances the overall integrity of the software development lifecycle.

### Disadvantages:

Advanced security features are often limited to paid plans. Implementing security practices can be complex and time-consuming, and managing security configurations across large teams or organizations may require additional expertise.

### Types:

1. **Dependency Scanning:** Identifies vulnerabilities in project dependencies.
2. **Secret Scanning:** Detects and alerts on leaked secrets like API keys.
3. **Code Scanning:** Analyzes code for potential security issues.
4. **Branch Protection:** Enforces rules to protect important branches from unauthorized changes.
5. **Two-Factor Authentication (2FA):** Adds an extra layer of security to user accounts.
6. **Security Advisories:** Allows repository maintainers to privately discuss and publish information about security vulnerabilities.

### Working:

GitHub Security tools continuously monitor repositories for vulnerabilities, secrets, and code issues. When a potential risk is detected, alerts are generated, allowing developers to take action. Branch protection rules can be set to enforce security policies, such as requiring code reviews before merging. Two-factor authentication (2FA) adds an extra verification step for user logins, while role-based access control (RBAC) limits access to sensitive parts of the repository. Security advisories enable private discussions and coordinated disclosure of vulnerabilities, ensuring responsible handling of security issues.

## Securing GitHub Account using Dependency Scanning

### Definition:

Dependency scanning checks for known vulnerabilities in your project dependencies. GitHub can identify and alert you to security risks in third-party libraries you use. It helps you proactively mitigate potential security issues in your code.

### Types:

1. **Vulnerability:** A security flaw that could be exploited.
2. **Severity:** The potential impact of the vulnerability.
3. **Fix:** Recommended actions to address the vulnerability.

### Configuration:

1. **Navigate to Your Repository:** Go to the main page of your repository on GitHub.
2. **Access Settings:** Click on the Settings tab.
3. **Security & Analysis:** In the sidebar, click on Code security and analysis under the Security section.
4. **Enable Dependency Graph:** Ensure the Dependency graph is enabled. If not, click Enable.
5. **Enable Dependabot Alerts:** Under Dependabot alerts, click Enable.
6. **Enable Dependabot Security Updates:** Under Dependabot security updates, click Enable.
7. **Configure Dependabot (Optional):** If you want to customize Dependabot updates, create or edit the .github/dependabot.yml file in your repository.

## Securing GitHub Account using Secret Scanning

### Definition:

Secret scanning detects sensitive information like API keys or passwords in your code. GitHub analyzes your repositories for potentially exposed secrets and alerts you to them. It helps prevent accidental exposure of confidential information and maintains security.

### Configuration:

1. **Navigate to Your Repository:** Go to the main page of your repository on GitHub.
2. **Access Settings:** Click on the Settings tab.
3. **Security & Analysis:** In the sidebar, click on Code security and analysis under the Security section.
4. **Enable GitHub Advanced Security:** Check if GitHub Advanced Security is enabled. If not, click Enable.
5. **Enable Secret Scanning:** Scroll down to the Secret scanning section and click Enable.
6. **Enable Push Protection (Optional):** If you want to prevent secrets from being pushed to the repository, enable Push protection.

## Securing GitHub Account using Code Scanning

### Definition:

Code scanning analyzes your code for potential security vulnerabilities. GitHub uses static analysis tools to detect common security issues. It helps identify and fix vulnerabilities before they can be exploited.

### Types:

Code Scanning can detect a wide range of vulnerabilities, including SQL injection, cross-site scripting (XSS), and buffer overflows.

### Configuration:

1. **Navigate to Your Repository:** Go to the main page of your repository on GitHub.
2. **Access Settings:** Click on the Settings tab.
3. **Security & Analysis:** In the sidebar, click on Code security and analysis under the Security section.
4. **Enable Code Scanning:** Click on Set up under Code scanning.
5. **Choose a Tool:** Select CodeQL or another code scanning tool.
6. **Configure Scanning:** Follow the prompts to configure the scanning tool and set up the workflow.
7. **Run Scans:** Save the configuration and run the scans to start detecting vulnerabilities.

## Securing GitHub Account using Branch Protection

### Definition:

Branch protection safeguards your critical branches from unauthorized changes. You can enforce rules like requiring code reviews or preventing direct pushes to specific branches. This helps ensure the stability and integrity of your codebase.

### Configuration:

1. **Navigate to Your Repository:** Go to the main page of your repository on GitHub.
2. **Access Settings:** Click on the Settings tab.
3. **Branches:** In the sidebar, click on Branches under the Code and automation section.
4. **Add Branch Protection Rule:** Click on Add rule next to Branch protection rules.
5. **Specify Branch Name Pattern:** Enter the branch name or pattern you want to protect.
6. **Configure Protection Rules:** Choose the rules you want to enforce, such as requiring pull request reviews, status checks, or signed commits.
7. **Save Changes:** Click Create to save the branch protection rule.

## Securing GitHub Account using 2FA

### Definition:

Two-Factor Authentication (2FA) adds an extra layer of security to your GitHub account. It requires not only your password but also a second factor, like a code from your phone, to log in. This helps protect your account even if your password is compromised.

### Configuration:

1. **Navigate to Your Profile:** Click on your profile photo in the upper-right corner and select Settings.
2. **Access Security Settings:** In the sidebar, click on Password and authentication under the Access section.
3. **Enable 2FA:** In the Two-factor authentication section, click Enable two-factor authentication.
4. **Choose Authentication Method:** Select either a TOTP app (like Google Authenticator) or SMS for receiving codes.
5. **Set Up and Verify:** Follow the prompts to set up your chosen method and verify it by entering the code sent to your app or phone.
6. **Save Recovery Codes:** Download or print your recovery codes and store them in a safe place.

## Securing GitHub Account using PassKeys

### Definition:

PassKeys are a modern and secure method of authentication that eliminates the need for traditional passwords. Instead, PassKeys use public-key cryptography to allow you to log in using biometric data (like fingerprints or facial recognition) or a hardware key. This makes accessing your GitHub account both easier and more secure, reducing the risk of phishing and password theft.

### Configuration:

1. **Navigate to Your Profile:** Click on your profile photo in the upper-right corner and select Settings.
2. **Access Security Settings:** In the sidebar, click on Password and authentication under the Access section.
3. **Enable PassKeys:** In the Passkey section, click on Set up passkey.
4. **Choose Authentication Device:** Select a device that supports PassKeys, such as a hardware security key or a device with biometric capabilities (fingerprint, facial recognition).
5. **Set Up and Verify:** Follow the prompts to register your PassKey. If using a device with biometrics, authenticate with your fingerprint or face. If using a hardware key, insert the key and follow instructions to verify.
6. **Save Backup Method:** Ensure you have a backup authentication method (e.g., recovery codes or an additional hardware key) in case you lose access to your primary device.

## Securing GitHub Account using GPG

### Definition:

GPG (GNU Privacy Guard) keys are used to sign Git commits and tags, ensuring that the changes you make in your GitHub repository are verified and trusted. By using a GPG key, you can confirm the authenticity of your commits, adding an extra layer of security and trust to your GitHub contributions.

### Configuration:

1. **Navigate to Your Profile:** Click on your profile photo in the upper-right corner and select Settings.
2. **Access SSH and GPG Keys:** In the sidebar, click on SSH and GPG keys under the Access section.
3. **Add a GPG Key:** Scroll to the GPG keys section and click New GPG key.
4. Generate or Import GPG Key:
5. If you don’t have a GPG key, use a GPG tool to generate one (e.g., gpg --full-generate-key on your local machine).
6. Once generated, export your public key using gpg --armor --export your\_email@example.com and copy it.
7. **Add GPG Key to GitHub:** Paste the GPG key into the text field and click Add GPG key.
8. **Set Up Verification in Git:** Configure Git to use your GPG key for signing commits by running git config --global user.signingkey <your-gpg-key-id>.
9. **Save Changes:** Your GPG key is now associated with your GitHub account, and commits you sign will be verified as trusted.

## GitHub Support

1. **Email Support:** Users can reach out to the GitHub support team via email for assistance with various issues or questions.
2. **Live Chat:** GitHub offers a live chat feature, providing users with real-time support and the ability to get immediate answers.
3. **Self Service:** The comprehensive GitHub Docs can often provide users with the information they need to resolve issues on their own.
4. **Community Forum:** Users can also engage with the wider GitHub community to share insights, ask questions, and find solutions.

## GitHub Workflow

### Definition:

GitHub Workflow is an automated process defined within a GitHub repository that allows developers to build, test, and deploy their code. It is typically configured using YAML files and leverages GitHub Actions to automate various development and operational tasks.

### Features:

Includes the ability to define custom CI/CD pipelines, automate tasks like testing, deployment, and code scanning, integrate with external tools and services, trigger workflows based on events (e.g., push, pull request), and manage workflow runs with logs and status checks.

### Advantages:

Automates repetitive tasks, improving efficiency and consistency in the development process. It integrates seamlessly with GitHub repositories, provides flexibility to create complex automation scenarios, and enhances collaboration by ensuring code quality and reducing manual errors.

### Disadvantages:

Can be complex to set up, especially for more advanced workflows, requiring knowledge of YAML and GitHub Actions. Debugging workflow failures may be challenging, and certain advanced features may be restricted to paid plans or require additional configuration.

### Working:

1. **Fork the Repository:** Create a copy of the original repository to work on your changes.
2. **Create a Branch:** Branch off the main branch to isolate your changes and avoid affecting the original codebase.
3. **Make Changes:** Make your changes to the code in your branch, test, and commit your changes.
4. **Push Changes:** Push your changes to your forked repository.
5. **Open a Pull Request:** Request to merge your branch into the main branch of the original repository.
6. **Review and Merge:** Other developers will review your changes, provide feedback, and approve the pull request.
7. **Deploy Changes:** Once merged, the changes will be deployed to production.

## GitHub Dashboard Walkthrough

Central hub for managing GitHub activities and tracking updates across repositories. Includes shortcuts to your repositories, recent activity, and announcements. The dashboard provides a centralized view of your GitHub activities, including your repositories, issues, and pull requests.

1. **Search Bar:** Allows searching for repositories, code, issues, pull requests, users, and organizations across GitHub. You can refine search results by specific repositories, users, or types of files.
2. **Create New (Plus Icon):**
3. New Repository: Start a new repository with options to initialize with a README, .gitignore, and license.
4. Import Repository: Import existing repositories from other version control systems.
5. New Codespace: Create a cloud-hosted development environment for coding directly on GitHub.
6. New Gist: Share snippets of code or text in a public or secret gist.
7. New Organization: Create a shared account for collaborative projects with multiple users.
8. New Project: Set up a project board to manage issues, pull requests, and tasks.
9. **Issues:** Access and manage issues assigned to or created by you across all repositories.
10. **Pull Requests:** Track, review, and manage pull requests that you are involved with.
11. **Notifications:** View and manage notifications related to repositories, issues, pull requests, and discussions.
12. **Top Repositories:** Displays the most active or starred repositories you are working on or following.
13. **New:** Create a new repository with options to initialize with a README, .gitignore, and license
14. **Home:** Quick access to your feed, issues, pull requests, and more.
15. **Send Feedback:**
16. Report a Bug: Notify GitHub of any bugs or technical issues you encounter on the platform.
17. Request a Feature: Suggest new features or improvements that you’d like to see implemented.
18. General Feedback: Share your thoughts, opinions, or experiences using GitHub.
19. **Filter (Activity Feed):**
20. Events: Choose specific events like pushes, issues, and pull requests to track.
21. Activity Feed Customization: Tailor the types of activities displayed on your feed, such as commits or comments.
22. Announcements: Display official GitHub announcements.
23. Special Discussion Posts: Highlight specific posts from discussions in repositories.
24. Releases: Track new releases from repositories you follow.
25. Update Posts: Display update posts from repositories.
26. Sponsors: Information on sponsorships related to projects or developers.
27. Stars: Track activities related to starred repositories.
28. Repositories: Monitor repositories created, forked, or starred by people you follow.
29. Repository Activity: View activities from repositories you are following, including issues and pull requests.
30. Follows: Track who people you follow are following.
31. Recommendations: GitHub's suggestions for repositories or developers you might be interested in.
32. **Latest Changes:**
33. Release Notes: Detailed notes about recent updates or changes to the platform.
34. New Features: Highlights any new tools, functionalities, or UI changes that have been rolled out.
35. Bug Fixes: Information on bugs that have been fixed in the latest release.
36. Performance Improvements: Details on optimizations made to improve the speed and reliability of GitHub.
37. **Explore Repositories**
38. Trending Repositories: View repositories that are currently trending based on stars, forks, or activity.
39. Topics: Browse repositories grouped by specific topics like machine learning, web development, or DevOps.
40. Recommended Repositories: Personalized suggestions based on your interests, starred repositories, and activity.
41. Collections: Curated lists of repositories centered around specific themes or technologies
42. **---**
43. Explore: Discover new repositories, topics, and developers based on your interests.
44. Marketplace: Access tools and integrations that enhance your GitHub experience, such as CI/CD tools.
45. Repositories: List of repositories you own, contribute to, or have starred.
46. **Profile Icon:**
47. Account Switcher

* **Add Account:** Add another GitHub account to your session for easy switching.
* **Switch Accounts:** Quickly toggle between different GitHub accounts, each retaining its session and settings.
* **Manage Accounts:** Overview and management of all accounts linked to the account switcher, including removing or reordering accounts.

1. **Your Profile:**
2. **Set Status:** Update your current status (e.g., busy, available).
3. **Your Profile:** Manage and edit your personal GitHub profile.
4. **Your Repositories:** Access all your repositories.
5. **Your Copilot:** Manage GitHub Copilot, the AI coding assistant.
6. **Your Projects:** Manage project boards related to your account.
7. **Your Stars:** View repositories you’ve starred.
8. **Your Gists:** Manage gists you’ve created or starred.
9. **Your Organizations:** Manage and switch between organizations.
10. **Your Enterprises:** Access GitHub Enterprise accounts.
11. **Your Sponsors:** Manage sponsorships.
12. **Try Enterprise:** Explore options for GitHub Enterprise, tailored for organizations with advanced needs.
13. **Feature Preview:** Access and test new or experimental features GitHub is working on.
14. **Settings:** Configure account settings, security, notifications, and integrations.
15. **GitHub Docs:** Access comprehensive documentation on GitHub features, tools, and best practices.
16. **GitHub Support:** Reach out to GitHub support for help with any issues or questions.
17. **GitHub Community:** Engage with the wider GitHub community, participate in discussions, and find resources.
18. **Sign Out:** Log out of your GitHub account securely.

## GitHub Settings Walkthrough

1. **Public Profile:**
2. **Profile Picture:** Upload or change your profile picture.
3. **Name:** Set or update your display name.
4. **Public Email:** Choose the email address you want to display publicly on your profile.
5. **Bio:** Write a short description about yourself.
6. **Pronouns:** Specify your preferred pronouns.
7. **URL:** Provide a link to your personal or professional website.
8. **ORCID iD:** Enter your ORCID iD, which is a unique identifier for academic authors.

* Connect your ORCID iD: Link your ORCID iD to your GitHub account to showcase your academic or research work.

1. **Social Accounts:** Link to your Twitter, LinkedIn, or other social media accounts.
2. **Company:** Specify your company or organization.
3. **Location:** Add your geographic location.

* Display Current Local Time: Show your current local time on your profile for better communication with collaborators.

1. **Update Profile:** Save changes made to your public profile.
2. **Contributions & Activity:**
3. **Make profile private and hide activity:** Hide your activity from the public and only show contributions to private repositories.
4. **Include private contributions on my profile:** Display contributions to private repositories in your contribution graph.
5. **Update Preferences:** Save changes made to contribution and activity preferences.
6. **Profile Settings:**
7. **Show Achievements on my profile:** Display earned achievements, like badges and trophies, on your public profile.
8. **Update Preferences:** Save changes to your profile settings.
9. **GitHub Developer Program:**

Join and access resources for building, testing, and distributing GitHub integrations and apps.

1. **Jobs Profile:**
2. **Available for hire:** Indicate that you are open to new job opportunities.
3. **Save jobs profile:** Save changes made to your jobs profile.
4. **Trending Settings:**
5. **Preferred spoken language:** Select your preferred spoken language for recommendations on trending repositories and developers.
6. **Save Trending Settings:** Save changes to your trending settings and preferences.
7. **Account:**
8. **Change Username:**
9. **Change Username:** Update your GitHub username, which is used in your GitHub profile URL and repositories.
10. **Link Patreon Account:**
11. **Connect with Patreon:** Link your GitHub account to Patreon, allowing supporters to see your public projects and contributions.
12. **Export Account Data:**
13. **Start Export:** Initiate the process to export your account data. This may take some time, and GitHub will notify you when the export is ready.
14. **Successor Settings:**
15. **Search by username, full name, or email address:** Assign a successor to your GitHub account by selecting someone from your connections. A successor can manage your repositories and projects if needed.
16. **Delete Account:**
17. **Delete your account:** Confirm the deletion of your account, understanding that this action is irreversible, and all data will be permanently lost.
18. **Appearance:**
19. **Theme Preferences**
20. **Theme Mode:** Select the overall theme for your GitHub interface.
21. **Day Theme:** Choose the light mode for better visibility during daytime or bright environments.
22. **Night Theme:** Choose the dark mode for easier viewing in low-light or nighttime settings.
23. **Emoji Skin Tone Preference:**
24. **Preferred Default Emoji Skin Tone:** Choose the default skin tone for emojis (e.g., light, medium, dark) that GitHub will use when you insert emojis.
25. **Tab Size Preferences**
26. **Choose the Number of Spaces a Tab Is Equal to When Rendering Code:** Define the number of spaces for each tab, typically 2 or 4 spaces, depending on your coding style.
27. **Markdown Editor Preferences**
28. **Use a Fixed-Width (Monospace) Font When Editing Markdown:** Enable this option to display text in a monospace font (like a code editor) while editing Markdown for better readability.
29. **Accessibility**
30. **Keyboard Shortcuts:**
31. **Character Keys:** Enable or disable specific character keys for navigating GitHub more efficiently.
32. **Save Keyboard Shortcut Preferences:** Save the changes made to your keyboard shortcuts for easy navigation.
33. **Motion**
34. **Autoplay Animated Images:** Set whether animated images (like GIFs) should automatically play when viewed.

* **Sync with System:** Automatically adjust motion settings based on your system preferences.
* **Enabled:** Turn on the autoplay feature for animated images.
* **Disabled:** Disable autoplay to reduce motion or prevent distractions.

1. **Save Motion Preferences:** Save your changes for how motion is handled across GitHub.
2. **Content**
3. **Link Underlines:** Customize how hyperlinks are displayed in GitHub content.

* **Hide Link Underlines:** Remove underlines from links for a cleaner appearance.
* **Show Link Underlines:** Display underlines on links to improve accessibility and visibility.

1. **Save Content Preferences:** Save your preferred settings for how links are displayed.
2. **Hovercards:** View quick information about users, repositories, and issues by hovering over links.
3. **Save Hovercard Preferences:** Save your settings for hovercards, which show additional information when you hover over usernames, issues, or pull requests.
4. **Editor Settings**
5. **URL Paste Behavior:** Set how URLs are formatted when pasted into GitHub's editor.

* **Formatted Link:** Automatically convert pasted URLs into clickable links with markdown formatting.
* **Plain Text:** Paste URLs as plain text without formatting.

1. **Save Editor Settings:** Save your preferences for URL pasting and editor behavior.
2. **Notifications**
3. **Default Notifications Email:** Set the primary email address where GitHub will send all notifications.
4. **Custom Routing:** Manage and configure personalized paths for navigating between different sections or repositories on GitHub.
5. **Automatically Watch Repositories:** Automatically start watching repositories you contribute to, allowing you to receive notifications about updates.
6. **Automatically Watch Teams:** Automatically watch discussions and activity from teams you are a part of to stay informed.
7. **Subscriptions:**
8. **Watching:** View the list of repositories and discussions you are actively watching to receive updates.

* **Notify me: on GitHub, Email:** Choose whether to receive notifications for watched repositories either on GitHub, via email, or both.

1. **Participating, @mentions, and Custom:** Set notifications for repositories or issues where you are directly participating or mentioned.

* **Notify me: on GitHub, Email:** Choose where you want to receive notifications for participation or mentions, either on GitHub, via email, or both.

1. **Customize Email Updates:** Customize how frequently and what types of email notifications you receive for your activity on GitHub.

* **Reviews, Pushes, Comments:** Enable or disable email notifications for specific activities like code reviews, repository pushes, or comments.

1. **Ignored Repositories:** View and manage repositories you have chosen to ignore notifications from, ensuring you don’t receive unnecessary updates.
2. **System:**
3. **Actions:** Set preferences for notifications about GitHub Actions.

* **Notify me: on GitHub, Email (Failed workflows only):** Choose to receive notifications about failed workflows only on GitHub or via email.

1. **Dependabot Alerts:** Receive notifications about new vulnerabilities found by Dependabot.

* **Notify me: on GitHub, Email, CLI:** Customize how you want to be notified for Dependabot alerts—via GitHub, email, or the CLI.

1. **Email Weekly Digest:** Opt to receive a weekly summary of Dependabot alerts instead of immediate notifications.
2. **'Deploy Key' Alert Email:** Receive specific email alerts regarding the use or changes of deploy keys in your repositories.
3. **Billing and Plans:**
4. **Plans and Usage:** View details about your current GitHub plan (e.g., Free, Pro, Team, Enterprise) and track your usage of features like GitHub Actions, Codespaces, and Packages.
5. **Spending Limits:** Set a cap on the amount you can spend on additional services, such as GitHub Actions minutes or Codespaces compute usage, to prevent overages.
6. **Payment Information:** Manage your payment methods, including credit card details or other billing information used for subscriptions and additional purchases on GitHub.
7. **Emails:**
8. **Primary Email:** Set your primary email address for GitHub communication.
9. **Add Emails:** Add and verify additional email addresses.
10. **Email Preferences:** Customize what types of email notifications you receive.
11. **Backup Email Address:** Add a secondary email address to ensure you can recover your account or receive important notifications if your primary email is inaccessible.
12. **Keep My Email Addresses Private:** Enable this option to hide your email addresses from public visibility, using a noreply email address for commits and other public activities.
13. **Block Command Line Pushes That Expose My Email:** Prevent accidental exposure of your private email address by blocking pushes from the command line that include your real email in commit metadata.
14. **Password and Authentication:**
15. **Password:**
16. **Change Password:** Update your account password.
17. **Passkeys:**
18. **Add a Passkeys:** Passwordless authentication method that uses biometric data (e.g., fingerprint, facial recognition) or a security key for secure and seamless logins to GitHub.
19. **Two-Factor Authentication (2FA):**
20. **Enable Two-Factor Authentication:** Enable or manage 2FA for added security.
21. **Sessions:**
22. **Web Sessions:** Manage and view active sessions on your GitHub account from various web browsers or devices. You can review recent logins, end any unwanted sessions, and ensure your account's security.
23. **GitHub Mobile Sessions:** View and manage active sessions on the GitHub mobile app. Monitor your account's activity on mobile devices and log out of any sessions you no longer recognize or use.
24. **SSH and GPG Keys:**
25. **SSH Keys:**
26. **New SSH Keys:** Add, view, or delete SSH keys for secure command-line access to your repositories.
27. **GPG Keys:**
28. **New CPG Keys:** Manage GPG keys for signing commits and tags, enhancing security and trust.
29. **Vigilant Mode:**
30. **Flag Unsigned Commits as Unverified:** Automatically mark commits without GPG signatures as unverified.
31. **Organizations:**
32. **New Organization:** Create a new GitHub organization to manage teams and projects collaboratively.
33. **Transform Account:** Convert your personal GitHub account into an organization or enterprise account.
34. **Enterprises:**
35. **Enterprise Access:** View and manage the enterprises you belong to.
36. **Permissions and Roles:** Adjust roles and permissions for enterprise resources.
37. **Billing Information:** Manage billing and subscriptions for enterprise plans.
38. **Moderation:**
39. **Blocked Users:** View and manage users you've blocked from interacting with your GitHub account.
40. **Interaction Limits:** Set limits on interactions and activity to manage engagement and reduce spam.
41. **Code Review Limits:** Configure limits for code review requests and approvals to streamline review processes.
42. **Repositories:**
43. **Repository Default Branch:** Set or change the default branch for your GitHub repository.
44. **Update:** Modify or update repository settings, branches, or other configurations.
45. **Repositories:**
46. **Repositories:** View, manage, and organize all your GitHub repositories.
47. **Deleted Repositories:** Access and restore recently deleted repositories before permanent removal.
48. **Codespaces:**
49. **Dotfiles:**
50. **Automatically install dotfiles:**
51. **Secrets:**
52. **Codespace user secrets:**
53. **New Secret:**
54. **GPG Verification:**
55. **Enable:**
56. **Settings Sync:**
57. **Enable:**
58. **Trusted repositories:**
59. **All Repositories:** Apply access or security configurations to all repositories.
60. **Selected Repositories:** Apply access or security controls to only specific repositories.
61. **Select Repositories:**
62. **Access and Security:**
63. **Disabled:**
64. **All Repositories:**
65. **Selected Repositories:**
66. **Editor Preference:** Choose your preferred code editor for development work.
67. **Visual Studio Code:** Use Visual Studio Code as your default development environment.
68. **Visual Studio Code for the Web:** Use the web version of Visual Studio Code for browser-based development.
69. **JetBrains Gateway:** Select JetBrains Gateway as your preferred development tool for GitHub.
70. **JupyterLab:** Use JupyterLab as your development environment, ideal for data science workflows.
71. **Default Idle Timeout:** Set the default time before an inactive codespace is automatically stopped.
72. **Default Retention Period:** Set the default period to retain a stopped codespace before deletion.
73. **Host Image Version Preference:** Select the preferred version of the host image for your codespaces.
74. **Stable:** Choose the stable host image version for consistent and reliable performance.
75. **Beta:** Opt for the beta host image version to access new features and updates.
76. **Region:** Define the geographical region where your codespace will run.
77. **Set Automatically:** Allow GitHub to automatically choose the best region for your codespace.
78. **Set Manually:** Manually select the region for your codespace based on your preferences.
79. **Packages:**
80. **Packages Permissions:**
81. **Default Package Setting:** Set default access and visibility for newly created packages.
82. **Inherit Access from Source Repository:** Automatically apply repository permissions to its packages.
83. **Save:**
84. **Deleted Packages:**
85. **Search Deleted Packages:**
86. **Co-Pilot:**
87. **GitHub Copilot:**
88. **Start Free Trial:** Begin a free trial of GitHub Copilot to explore its features.
89. **Get Copilot from an Organization:**
90. **Create an Organization:** Set up a new GitHub organization to manage projects and teams collaboratively.
91. **Pages**:
92. **Add a Domain:** Connect a custom domain to your GitHub Pages site for a personalized URL.
93. **Pages Setup:** Configure GitHub Pages to host static websites from your repositories.
94. **Custom Domains:** Add and manage custom domains for your GitHub Pages sites.
95. **Visibility Settings:** Control who can view your GitHub Pages sites.
96. **Saved Replies:**
97. **Add a saved reply:** Add canned responses to common issues or pull requests.
98. **Manage Replies:** Edit or delete saved replies as needed.
99. **Quick Access:** Use saved replies in discussions or pull requests to streamline communication.
100. **Code security and analysis:**
101. **User:**
102. **Push Protection for Yourself:** Enable protection to prevent pushing known vulnerabilities.
103. **Repositories:**
104. **Private Vulnerability Reporting:** Allow private reports of vulnerabilities found in your repositories.

* **Automatically Enable for New Public Repositories:** Enable private vulnerability reporting by default for new public repositories.

1. **Dependency Graph:** Visualize and track dependencies in your repositories.

* **Automatically Enable for New Private Repositories:** Automatically enable the dependency graph for new private repositories.

1. **Dependabot:**
2. **Dependabot Alerts:** Receive alerts for vulnerabilities in dependencies across repositories.

* **Automatically Enable for New Repositories:** Automatically activate Dependabot alerts for new repositories.

1. **Dependabot Security Updates:** Get automated pull requests to fix vulnerabilities in your dependencies.

* **Automatically Enable for New Repositories:** Automatically enable Dependabot security updates for new repositories.

1. **Grouped Security Updates:** Group multiple security updates into a single pull request.

* **Automatically Enable for New Repositories:** Automatically group security updates in new repositories.

1. **Dependabot on Actions Runners:** Run Dependabot checks on GitHub-hosted actions runners.
2. **Dependabot on Self-Hosted Runners:** Run Dependabot checks on your self-hosted runners.

* **Automatically Enable for New Repositories:** Enable Dependabot for self-hosted runners in new repositories.

1. **Applications:**
2. **Installed GitHub Apps:** View and manage GitHub Apps installed on your account or organization.
3. **Authorized GitHub Apps:** Review and manage GitHub Apps you've granted access to your account.
4. **Authorized OAuth Apps:** Manage OAuth apps you've authorized to access your GitHub account.
5. **SonarCloud:** Integrate and configure SonarCloud for continuous code quality and security analysis.

* **Configure:** Adjust settings and preferences for GitHub Apps or services like SonarCloud.

1. **Scheduled reminders:**
2. **Reminder Settings:** Set up and manage reminders for reviewing pull requests or issues.
3. **Schedule Frequency:** Choose how often and when you want to receive reminders.
4. **Notification Methods:** Select how you want to receive reminders (e.g., email, Slack).
5. **Security log:**
6. **Filters:** Apply filters to narrow down specific actions in the security log.
7. **Filter Audit Logs:** Refine audit logs to show only relevant events based on criteria.

* **Yesterday's Activity:** Review security events that occurred on your account the previous day.
* **Repository Management:** Track changes related to repository settings and access.
* **Billing Updates:** Monitor any changes or updates related to billing and payments.
* **Copilot Activity:** View logs of actions performed by GitHub Copilot in your account.
* **Personal Access Token Activity:** Track the creation, usage, and deletion of personal access tokens.
* **View Advanced Search Syntax:** Access detailed search syntax to refine log searches.

1. **Export:** Download and export security logs for external analysis or record-keeping.
2. **Sponsorship log:**
3. **Sponsorship Activity:** Track all activities related to GitHub Sponsors, including who is sponsoring you or whom you are sponsoring.
4. **Sponsorship Management:** Manage sponsorship tiers, amounts, and communications with sponsors.
5. **Developer Settings:**
6. **GitHub Apps:** Create and manage GitHub Apps, which can automate tasks or integrate with external services.
7. **OAuth Apps:** Register and manage OAuth applications that interact with the GitHub API.
8. **Personal Access Tokens:** Create and manage tokens for accessing the GitHub API.
9. **Fine-grained Tokens:** Manage personal access tokens with precise, scoped permissions for enhanced security.
10. **Tokens (Classic):** View and manage legacy personal access tokens with broader access permissions.

## GitHub Community

### Definition:

GitHub Community is a forum where GitHub users can ask questions, share knowledge, and collaborate on open-source projects.

### Features:

1. User-driven forums with discussions on Git, GitHub, and development.
2. GitHub staff participate in discussions.
3. Access to a wide range of topics and FAQs.

### Advantages:

1. Quick access to community support.
2. Learn from others’ experiences and expertise.
3. Engaging platform for collaboration and problem-solving.

### Disadvantages:

1. Response times may vary.
2. Information may not always be verified or accurate.

### Types:

1. **Discussions:** A collaborative space for asking questions and sharing ideas about a project.
2. **Support forums:** Platforms where users can ask questions and receive assistance from the community.

## GitHub Community Walkthrough

1. **Search:**
2. **Search Bar:** Allows you to search for discussions, questions, and topics within the GitHub Community forum.
3. **Search Filters:** Narrow your search results by filtering based on relevance, date, and categories (e.g., discussions, questions, or announcements).
4. **Sort By:**
5. **Relevance:** Sort results based on how closely they match your search query or interests.
6. **Newest:** Display the most recent discussions or posts.
7. **Oldest:** Show the oldest discussions first for historical context.
8. **Most Popular:** Sort by popularity, typically based on the number of views, comments, or interactions.
9. **Label:**
10. **Issue Labels:** Filter discussions by labels such as “Question,” “Idea,” “Feedback,” “Bug Report,” or other custom labels.
11. **Popular Labels:** Browse trending labels to find discussions related to specific topics or issues.
12. **Filter:**
13. **Category Filter:** Filter discussions by category, such as "General," "Ideas," "Announcements," or "Feedback."
14. **State Filter:** Show discussions based on their state, like "Open," "Closed," or "Answered."
15. **Author Filter:** Filter discussions by the author, allowing you to view posts from specific community members.
16. **New Discussion:**
17. **Start a New Discussion:** Create a new discussion thread to ask questions, share ideas, or provide feedback.
18. **Category Selection:** Choose the appropriate category (e.g., Feedback, General, or Support) for your discussion.
19. **Formatting Options:** Use Markdown to format your discussion with headings, code blocks, lists, and more.
20. **Categories:**
21. **General:** Discussions about anything related to GitHub or the community.
22. **Feedback:** Suggestions or feedback about GitHub products and services.
23. **Support:** Help and troubleshooting discussions for GitHub issues or challenges.
24. **Ideas:** Share new ideas or feature requests with the GitHub team and community.
25. **Announcements:** Official announcements and news from GitHub.
26. **Discussions:**
27. **View Ongoing Discussions:** See ongoing conversations from the community, sorted by relevance or recency.
28. **Participate in Discussions:** Comment on existing discussions, reply to questions, or provide feedback.
29. **Upvote/Like:** Give positive feedback by upvoting or liking posts to highlight helpful responses.
30. **Most Helpful:**
31. **Top Answer or Solution:** View responses marked as the most helpful or accepted solution for a question or discussion.
32. **Community Votes:** Sort discussions based on the number of votes for the most helpful or insightful response.
33. **Mark as Helpful:** If you're the discussion creator, you can mark a response as the most helpful to highlight the best solution.

## GitHub Docs

### Definition:

The official documentation site for GitHub, providing comprehensive guides, tutorials, and API references.

### Features:

1. Detailed guides on GitHub features.
2. API and developer documentation.
3. Searchable knowledge base.

### Advantages:

1. Official and authoritative source for GitHub information.
2. Regularly updated with new features.

### Disadvantages:

1. Can be overwhelming for beginners.
2. Some advanced topics may lack in-depth examples.

### Types:

1. **GitHub Docs:** Official documentation providing guidance on using GitHub and its features.
2. **API documentation:** Technical references for interacting with GitHub's API.

## GitHub Docs Walkthrough

1. **Versions:**
2. **Dropdown Menu:** Select from different GitHub versions such as "GitHub Free," "GitHub Pro," "GitHub Team," and "GitHub Enterprise."
3. **All Enterprise Server Releases**

* **Release Versions:** A list of all available GitHub Enterprise Server versions, from the most recent to older releases.
* **Release Notes:** Access release notes for each version, detailing new features, bug fixes, and improvements.
* **Download:** Option to download documentation or assets for the specific Enterprise Server version.

1. **About Versions**

* **Versioning Information:** Detailed explanation about how GitHub manages versions across its platforms and products (e.g., GitHub.com vs GitHub Enterprise Server).
* **Compatibility:** Information on which versions of GitHub Enterprise Server are compatible with different features.
* **Support and End-of-Life Policies:** Guidelines on how long versions are supported and the lifecycle of each GitHub version.

1. **Search:**
2. **Search Bar:** Search for specific topics, guides, or keywords across GitHub Docs.
3. **Filter Results:** Narrow down search results based on relevance or specific topics (e.g., API documentation, GitHub Actions, security).
4. **Search Suggestions:** Get auto-completed suggestions as you type to quickly find relevant documentation or topics.
5. **Select Language:**
6. **Language Options:** Choose from multiple languages to view GitHub documentation (e.g., English, Spanish, Japanese, etc.).
7. **Localized Content:** Access content translated into your selected language for easier reading and understanding.
8. **Fallback to English:** If certain documentation is not available in your selected language, the English version will be displayed by default.

## Feature Preview

### Definition:

A GitHub option that allows users to try and provide feedback on upcoming features before they are fully released.

### Features:

1. Early access to beta features.
2. User feedback directly influences final product.

### Advantages:

1. Opportunity to influence new GitHub features.
2. Test and familiarize yourself with new tools.

### Disadvantages:

1. Beta features may have bugs or incomplete functionality.
2. Frequent updates might cause disruptions.

### Types:

1. **Beta feature previews:** Early access to experimental features before official release.
2. **Private Previews:** Limited to specific users or organizations

## GitHub Feature Preview Walkthrough

1. **Colorblind Themes:** Customizable themes designed to enhance visibility for users with color vision deficiencies.
2. **Command Palette:** A quick-access interface for executing GitHub commands and navigating GitHub features.
3. **Rich Jupyter Notebook Diffs:** Enhanced diff view for Jupyter Notebooks, making it easier to review changes in notebook files.
4. **New Pull Request Commits Experience:** Updated user interface for viewing commits in pull requests.
5. **Enhanced Repos Insights Views:** Improved views and analytics for repository insights.
6. **Slash Commands:** Commands that can be executed within GitHub issues, pull requests, and discussions using a slash (/) prefix.

## Create Feature Preview

1. Navigate to the Feature Preview page by clicking on your profile picture.
2. Select Feature Preview from the dropdown.
3. Browse the list of available previews and click Enable next to the desired feature.
4. Confirm by reading the description and enabling it.

## Try Enterprise

### Definition:

A feature promoting GitHub Enterprise, offering users a trial experience to explore the advanced collaboration, security, and admin tools.

### Features:

1. Free trial of GitHub Enterprise.
2. Access to enhanced security features.
3. Advanced management and compliance tools.

### Advantages:

1. Helps organizations evaluate Enterprise features.
2. Free trial without commitment.

### Disadvantages:

1. Limited trial period.
2. Requires setup for evaluation.

### Types:

1. **Enterprise Cloud trial:** A trial period to explore GitHub Enterprise Cloud features for organizations.
2. **Enterprise Server trial:** A trial period to test GitHub Enterprise Server for self-hosted environments.

## GitHub Try Enterprise Walkthrough

1. **Enterprise with personal accounts:** Option to set up a GitHub Enterprise environment where users can use their personal GitHub accounts.
2. **Account Integration:** Users log in with their existing personal GitHub accounts while accessing enterprise features.
3. **Feature Access:** Enterprise features such as advanced security, analytics, and organization management are available to personal accounts.
4. **Administration:** Admins can manage users, access controls, and integrations while users maintain their personal accounts.
5. **Customization:** Customize enterprise settings to align with organizational policies while using personal GitHub accounts.
6. **Enterprise with managed users:** Set up a GitHub Enterprise environment where user accounts are managed and controlled by the enterprise.
7. **User Management:** Admins create and manage user accounts through an enterprise directory, such as LDAP or SAML.
8. **Access Control:** Centralized control over user permissions, roles, and access to repositories and features.
9. **Integration:** Integration with enterprise identity providers for user authentication and provisioning.
10. **Policy Enforcement:** Enforce organizational policies and compliance requirements with managed user accounts.

## Your Sponsors

### Definition:

GitHub Sponsors allows developers and organizations to financially support open-source contributors and projects.

### Features:

1. Direct financial support for developers.
2. Monthly sponsorship options.
3. Tiers for different levels of support.

### Advantages:

1. Encourages open-source development.
2. Offers a revenue stream for contributors.

### Disadvantages:

1. Limited visibility for smaller projects.
2. Requires active promotion to attract sponsors.

### Types:

1. **Individual Sponsors:** A way for users to financially support open-source developers directly. Recognition, exclusive content, and more. Showcased on your profile
2. **Organization Sponsors:** A method for organizations to sponsor open-source projects and maintainers. Branded sponsorship, custom perks, and more. Prominent placement on your profile
3. **GitHub Sponsors Program:** Financial support and growth opportunities. Amplified through GitHub's channels

## GitHub Sponsors Walkthrough

1. **Manage Who You Sponsor**
2. **Sponsorship Tiers:** View and modify the sponsorship tiers you are contributing to, including adjusting the amount or canceling sponsorships.
3. **Sponsorship History:** Track your past sponsorships, including start and end dates, along with total contributions.
4. **Payment Management:** Update your payment methods and billing details related to your sponsorships.
5. **GitHub Sponsors Eligible Accounts**
6. **Eligibility Criteria:** Learn about the requirements for developers and organizations to become eligible for GitHub Sponsors (e.g., contributing to open source).
7. **Discover Eligible Accounts:** Find and explore profiles of eligible developers and projects that you can sponsor.
8. **Sponsorship Tiers:** View the available sponsorship tiers for eligible accounts and choose the level of contribution that suits you.
9. **Get Sponsors**
10. **Application Process:** Information on how to apply for GitHub Sponsors and the steps to get approved as a sponsored developer.
11. **Profile Setup:** Customize your sponsorship profile, including adding sponsorship tiers, setting goals, and writing a description of your work.
12. **Payout Settings:** Configure payout details such as linking your bank account or payment method to receive sponsorship funds.
13. **Promote Sponsorships:** Learn tips on promoting your sponsorship profile to attract sponsors through social media or GitHub’s discoverable features.

## Your Enterprises

### Definition:

GitHub Enterprise offers additional features tailored for large organizations, including security, compliance, and advanced collaboration tools.

### Features:

1. Enhanced security and compliance.
2. Centralized administration tools.
3. Integration with enterprise tools.

### Advantages:

1. Scales for large organizations.
2. Supports enterprise-level collaboration and management.

### Disadvantages:

1. Higher cost.
2. Requires more configuration and maintenance.

### Types:

1. **GitHub Enterprise Cloud:** A cloud-based version of GitHub tailored for businesses with advanced security and administrative features.
2. **GitHub Enterprise Server:** A self-hosted version of GitHub designed for enterprises with strict compliance or security requirements.

## GitHub Enterprises Walkthrough

1. **Enterprises**
2. **Enterprise Overview:** View details about your enterprise plan, including the number of organizations, users, and overall activity.
3. **Access Controls:** Manage enterprise-wide settings, including user permissions, SAML configurations, and security policies.
4. **Organization Management:** Add, remove, or manage the organizations linked to your enterprise account.
5. **Billing and Usage:** Access detailed billing information and track usage metrics for your enterprise plan.
6. **Enterprise Insights:** Analyze insights such as contributions, pull requests, and repository statistics across all linked organizations.
7. **Start Free Trial**
8. **Trial Duration:** Information on the trial period (e.g., 30 days) and what features are available during the trial.
9. **Signup Process:** Quick steps to sign up for the free trial, including creating or linking an enterprise account.
10. **Feature Access:** During the trial, access advanced enterprise features such as security, compliance, team management, and analytics.
11. **Upgrade Option:** Once the trial ends, easily upgrade to a paid enterprise plan with options for continued usage.

## Your Organizations

### Definition:

Organizations allow groups of people to collaborate on GitHub repositories under a shared workspace with custom permissions.

### Features:

1. Centralized management of repositories.
2. Team-based permissions.
3. Shared billing.

### Advantages:

1. Facilitates collaboration across teams.
2. Flexible permissions and access control.
3. Centralized code management for large projects.

### Disadvantages:

1. Requires proper management of permissions.
2. Can become complicated with many repositories and users.

### Types:

1. **Public organization:** A GitHub organization whose repositories and members are visible to the public.
2. **Private organization:** A GitHub organization with restricted access to its repositories and members.

## GitHub Organizations Walkthrough

1. **Organizations**
2. **Organization List:** View all organizations you belong to or have administrative rights over.
3. **Settings Access:** Quickly access organization settings such as member management, repository permissions, and billing.
4. **Activity Overview:** Monitor recent activity across your organizations, including contributions, issues, and pull requests.
5. **Invite Members:** Add new members to your organization and assign roles such as admin, member, or billing manager.
6. **New Organizations**
7. **Creation Process:** Step-by-step guidance for setting up a new organization, including naming, team setup, and repository management.
8. **Plan Selection:** Choose from different GitHub plans (Free, Pro, Team, Enterprise) for your new organization.
9. **Team and Permissions:** Set up teams within the organization and configure permissions for collaboration on repositories.
10. **Transform Account**
11. **Conversion Benefits:** Learn the advantages of transforming an account, such as enhanced team collaboration, access control, and organization-level management features.
12. **Steps to Transform:** Follow the steps to transfer repositories, teams, and projects from a personal account to an organization.
13. **Turn Into an Organization**
14. **Preserve Repositories:** Safeguard repositories, projects, and settings while transforming the personal account into an organization.
15. **Team Collaboration:** Gain the ability to invite members, manage teams, and set permissions at the organization level after transformation.
16. **Ownership Transfer:** Reassign the ownership of repositories to ensure proper management after the account is turned into an organization.

## Create organization

1. Navigate to GitHub.
2. Click your profile icon and select Your organizations.
3. Click New Organization.
4. Choose a plan (Free, Team, Enterprise).
5. Name your organization and associate an email.
6. Complete the setup and click Create.

## Your Gists

### Definition:

GitHub Gists allow users to share code snippets or small projects, either publicly or privately.

### Features:

1. Instant code sharing.
2. Public and secret gists.
3. Support for version control.

### Advantages:

1. Easy sharing of code snippets.
2. Gists can be forked or cloned like repositories.

### Disadvantages:

1. Limited functionality compared to full repositories.
2. Secret gists are not fully private.

### Types:

1. **Public gist:** A snippet of code or text that is publicly accessible on GitHub.
2. **Secret gist:** A gist that is only accessible via a unique URL and not listed publicly.

## Creating a GitHub Gist

1. Go to gist.github.com.
2. Click New Gist.
3. Add a file name and description.
4. Paste your code or text.
5. Choose between Public or Secret gist.
6. Click Create Secret Gist or Create Public Gist.

## Forking and Cloning a Gist

1. Navigate to the gist.
2. Click the Fork button in the top right.
3. To clone, click Clone via SSH or HTTPS under the gist, then run the Git command to clone it locally.

## Your Stars

### Definition:

Stars allow users to bookmark repositories they find useful or interesting, creating a personal collection of favorite repositories.

### Features:

1. Quick access to starred repositories.
2. Ability to explore popular repositories starred by others.
3. Organize stars into lists or categories.

### Advantages:

1. Simple way to bookmark projects.
2. Helps discover trending repositories.

### Disadvantages:

1. No built-in categorization or tagging system.
2. Managing many stars can become overwhelming.

### Types:

1. **Starred repositories:** Repositories that a user has bookmarked for future reference.

## Saving a Repository with Stars

1. Navigate to the repository page.
2. Click the Star button in the top-right corner to save it.
3. To view your starred repositories, go to your profile and click on the Stars tab.

## Your Projects

### Definition:

GitHub Projects provide a way to organize tasks, issues, and pull requests for a repository, improving project management and workflow.

### Features:

1. Kanban-style boards.
2. Link issues and pull requests to tasks.
3. Customizable columns for workflows.

### Advantages:

1. Visual organization of work.
2. Helps manage complex projects.
3. Integration with GitHub issues and pull requests.

### Disadvantages:

1. Basic compared to dedicated project management tools.
2. Limited reporting capabilities.

### Types:

1. **Classic Projects:** GitHub's older project management feature for organizing tasks in a Kanban-style board.
2. **GitHub Projects Beta (enhanced features):** The new version of GitHub Projects with advanced tracking and collaboration features.

### Project Layout:

1. **Classic:** Simple and traditional Kanban board layout.
2. **List:** Displays tasks in a list format, ideal for project planning and overview.
3. **Timeline:** Visualizes project progress in a timeline view, showing deadlines and dependencies.

### Project Tools:

1. **Kanban:** Visualize and manage your project tasks using a Kanban-style board, enabling efficient workflow and project tracking.
2. **Calendar:** Stay on top of deadlines and important project milestones with the built-in calendar view, helping you plan and coordinate effectively.
3. **Analytics:** Leverage the project analytics to gain insights into your team's productivity, task progress, and overall project health.
4. **Automation:** Automate repetitive tasks and workflows within your projects, streamlining your project management processes and saving valuable time.

### Project Workflow:

1. **Define Stage:** Create a visual representation of your project's workflow, breaking it down into clear stages.
2. **Automate Tasks:** Integrate automation into your workflow to streamline repetitive tasks and save time.
3. **Track Program:** Monitor the progress of your project through each stage and identify potential bottlenecks.
4. **Improve Efficiency:** Streamline your workflow, reduce errors, and ensure a consistent and efficient approach.

### Project Configuration Options:

1. **Custom Columns:** Define custom columns to represent specific stages or workflows in your project.
2. **Filtering and Sorting:** Apply filters and sorting rules to organize and prioritize tasks based on different criteria.
3. **Automations:** Automate workflows by setting up rules to move tasks between columns or trigger actions.
4. **Notifications:** Receive notifications when tasks are updated, deadlines approach, or new issues are created.

## Creating a project

1. Go to Projects in the repository.
2. Click New Project and follow the prompts.

## Setting your project description and README

1. Inside the project, click Edit to add a description and a README file.

## Adding issues to your project

1. In your project board, click Add Item.
2. Select Issues from the list and link them to your project.

## Adding draft issues to your project

1. In your project board, click Add Draft.
2. Fill in the issue details and save.

## Adding an iteration field

1. In your project, click Fields.
2. Add a custom field for iteration tracking.

## Creating a field to track priority

1. Click Add Field in your project.
2. Set it up as a priority tracking field.

## Grouping issues by priority

1. In the project board, click Group By.
2. Select Priority from the dropdown.

## Saving the priority view

1. Once grouped by priority, click Save View to keep the layout.

## Adding a board layout

1. In your project, click Layout.
2. Choose the board layout to visualize tasks as cards.

## Configure built-in automation

1. Go to your project.
2. Select Automation and configure actions like auto-assign, close issues on PR merge, etc.

## Your Copilot

### Definition:

GitHub Copilot is an AI-powered coding assistant that helps users by suggesting code snippets, functions, and even entire blocks of code.

### Features:

1. AI-based code suggestions.
2. Works with various programming languages.
3. Real-time code generation as you type.

### Advantages:

1. Speeds up coding by providing relevant suggestions.
2. Reduces time spent on repetitive coding tasks.
3. Helps explore new coding patterns.

### Disadvantages:

1. AI suggestions might not always be accurate.
2. Can lead to over-reliance on AI-generated code.

### Types:

1. **Copilot for Individuals:** GitHub Copilot service available for individual developers to assist with code suggestions.
2. **Copilot for Business:** GitHub Copilot service tailored for business environments with team-based collaboration and control.

## Getting started with GitHub Copilot

1. Enable GitHub Copilot by navigating to your profile, and selecting Settings > Copilot.
2. Click Enable GitHub Copilot and follow the instructions.
3. Open VS Code and start typing; Copilot will start suggesting code.

## Your Repositories

### Definition:

A collection of project files and their history managed in GitHub, where developers can track and collaborate on code.

### Features:

1. Git version control integration.
2. Public and private repositories.
3. Collaboration features (issues, pull requests).
4. Security and vulnerability checks.

### Advantages:

1. Easy collaboration and version control.
2. Branching and merging support.
3. Visibility of project history and changes.

### Disadvantages:

1. Managing multiple repositories can become complex.
2. Conflicts may arise during merges.

### Types:

1. **Public repository:** A repository that is accessible to anyone on GitHub.
2. **Private repository:** A repository that is restricted to specific users or teams.

### Access Permissions:

1. **Read:** View project information and content.
2. **Write:** Edit project information and content.
3. **Admin:** Manage project settings and permissions.

### Visibility Options:

1. **Public:** Visible to everyone, including anonymous users.
2. **Private:** Only accessible to members with permissions. Requires a GitHub Pro, Team, or Enterprise plan.
3. **Internal:** Visible to all members of an organization. Requires a GitHub Team or Enterprise plan.
4. **Achieved:** Repository is read-only. Users can view content but cannot make changes.

### Components:

1. **Branches:** Branches allow you to work on different versions of your code simultaneously, keeping your main branch clean and stable.
2. **Pull Requests:** Pull requests are used to propose changes to the main branch, enabling collaborative code review and feedback.
3. **Issues:** Issues serve as a central hub for tracking bugs, feature requests, and other project tasks.

### Privacy Settings:

1. **Branch Protection:** Restrict changes to certain branches, requiring approvals or status checks.
2. **Codeowner:** Designate specific users or teams responsible for reviewing code changes to particular files or directories.
3. **Required Reviewer:** Require a specific number of reviewers or specific individuals to approve code changes before they can be merged.

### Repository Templates:

1. **Basic:** A simple template with a README, LICENSE, and a basic file structure.
2. **Project:** A template specifically designed for projects, often including a CONTRIBUTING file and more detailed instructions.
3. **Package:** A template tailored for software packages, including build scripts, testing frameworks, and documentation.

## GitHub Repositories Walkthrough

1. **Code**
2. **Repository Files:** View, navigate, and manage all files and folders in your repository.
3. **File History:** View the version history and commit logs for each file.
4. **Edit/Preview:** Make changes to files directly in the browser or preview markdown and code changes.
5. **Issues**
6. **Filters:** Apply filters to narrow down the list of issues and pull requests.
7. **Filter Issues:**

* **Open Issues and Pull Requests:** View all open issues and pull requests for the repository.
* **Your Issues:** View all issues that you have opened in the repository.
* **Your Pull Requests:** View all pull requests that you’ve opened in the repository.
* **Everything Assigned to You:** View all issues and pull requests assigned to you.
* **Everything Mentioning You:** See all issues and pull requests where you've been mentioned.
* **View Advanced Search Syntax:** Learn advanced filtering options using GitHub’s search syntax.

1. **Search:** Use specific operators to find issues (e.g., by label, status, milestone, author).
2. **Labels:** Organize and categorize issues with labels.
3. **Milestones:** Group issues and pull requests under milestones.
4. **New Issue:** Create a new issue in the repository.
5. **Open:** View all open issues in the repository.
6. **Close:** View all closed issues in the repository.
7. **Author:** Filter issues based on who created them.

* **Author Filter:** See all issues opened by a specific contributor or team member.

1. **Label:** Filter issues by their assigned labels.

* **Label Categories:** Sort issues based on tags like "bug," "enhancement," or "help wanted."

1. **Projects:** Link issues to specific GitHub Projects.

* **Project Management:** Track issues within project boards to organize them based on the project’s workflow.

1. **Milestones:** Filter issues based on assigned milestones.

* **Milestone Progress:** Track issues as part of a larger goal or project phase.

1. **Assignee:** Filter issues by assignee.

* **Assignee Focus:** See all issues assigned to a specific person or collaborator.

1. **Sort:** Sort the list of issues based on different criteria.

* **Sorting Options:** Sort by newest, oldest, most commented, least commented, etc.

1. **Pull Requests**
2. **Create Pull Request:** Submit changes from a branch to the base branch for review.
3. **Review Changes:** Comment on, approve, or request changes on code before merging.
4. **Merge Options:** Use different merge strategies like merge, squash, or rebase.
5. **Actions**
6. **Workflow Setup:** Create or configure CI/CD pipelines directly within the repository.
7. **View Workflow Runs:** Monitor the status of running workflows, including logs and errors.
8. **Actions Marketplace:** Browse and use pre-built actions from the GitHub Marketplace.
9. **Projects**
10. **Create Projects:** Set up new project boards to track issues, pull requests, and tasks.
11. **Project Cards:** Move issues and tasks across columns to track progress.
12. **Customizable Columns:** Add, remove, or rename columns to suit your workflow.
13. **Wiki**
14. **Create Pages:** Add new wiki pages to describe features, processes, or project information.
15. **Wiki History:** View and revert to earlier versions of wiki pages.
16. **Organize Wiki:** Structure wiki pages using categories or links for easy navigation.
17. **Security**
18. **Reporting:**

* **Policy:** Set security policies for the repository.
* **Advisories**: Manage and create security advisories for your repository.

1. **Vulnerability Alerts:**

* **Dependabot:** Tool to automatically keep dependencies up-to-date.
* **Code Scanning:** Analyze code for vulnerabilities and errors.
* **Secret Scanning:** Scans code to detect exposed secrets like API keys or tokens.

1. **Security Overview:** Dashboard summarizing the security status of your repository.
2. **Security Policy:** Define the security policy for handling vulnerabilities in your repository.
3. **Security Advisories:** Publish security advisories to inform users of vulnerabilities in your project.
4. **Private Sharing:** Option to privately disclose vulnerabilities before public disclosure.
5. **Private Vulnerability Reporting:** Mechanism for securely reporting vulnerabilities in private repositories.
6. **Dependabot Alerts:** Receive alerts when your dependencies have vulnerabilities.
7. **Code Scanning Alerts • Needs Setup:** Notifications for issues found during code scanning.
8. **Secret Scanning Alerts • Enabled:** Secret scanning is enabled and actively monitoring the repository.
9. **Insights**
10. **Pulse**: Shows recent activity in the repository.
11. **Contributors**: Shows a list of contributors to the repository.
12. **Community**: Shows how engaged the community is with the project.
13. **Community Standards**: A checklist of recommended community standards.
14. **Traffic**: Provides metrics about the traffic to the repository.
15. **Commits**: Displays commit history for the repository.
16. **Code Frequency**: Tracks how the codebase changes over time.
17. **Dependency Graph**: Shows a visual map of the repository’s dependencies.
18. **Network**: Visual representation of forks and clones of the repository.
19. **Forks**: Lists all the forks of the repository.
20. **Overview**: A general summary of the repository’s activity and contributions.
21. **Period**: Select a time period to view activity and insights over that timeframe.
22. **Filter Activity**: Filter repository activity to show specific types of events.

* **24 Hours**: Shows activity and insights from the last 24 hours.
* **3 Days**: Displays activity from the past three days.
* **1 Week**: Shows a week's worth of activity in the repository.
* **1 Month**: Displays activity from the past month.

1. **Settings**
2. **General**
3. **Repository Name:** Displays the current repository name.

* **Rename:** Allows you to rename the repository.

1. **Template Repository:** Option to mark the repository as a template for others to fork from.
2. **Require Sign-off on Web-Based Commits:** Enables contributors to sign off when making commits through the web interface.
3. **Default Branch**
4. **Main:** Displays the default branch.
5. **Rename:** Allows renaming of the default branch.
6. **Switch Branch:** Option to switch the default branch to another branch.
7. **Social Preview**
8. **Download Template:** Download an image template for the social preview of the repository.
9. **Edit:** Modify the social preview image that appears when shared on social media.
10. **Features**

* **Wikis:**
* **Restrict Wiki Editing:** Restrict wiki editing to collaborators only.
* **Issues:**
* **Set up Templates:** Create issue templates to streamline issue reporting.
* Sponsorships
* **Display Sponsor Button:** Option to display a “Sponsor” button on the repository for funding contributions.
* **Preserve Repository:** Keep the repository active for sponsorship purposes, even if it becomes archived.
* **Discussions**
* **Get Started with Discussions:** Enable Discussions for community interactions.
* **Set Up Discussions:** Configure Discussions settings and templates for the repository.
* **Projects**

1. **Pull Requests**

* **Allow Merge Commits:** Enable or disable the use of merge commits for pull requests.
* **Default Commit Message:** Set a default commit message for merge commits (pull request title or pull request title and description).
* **Allow Squash Merging**
* **Default Commit Message:** Set default commit messages for squash merges (pull request title or pull request title and commit details).
* **Allow Rebase Merging:** Add all commits from the head branch onto the base branch individually.
* **Always Suggest Updating Branches:** Option to suggest updating pull request branches before merging.
* **Allow Auto-Merge:** Enable automatic merging of pull requests when all required conditions are met.
* **Automatically Delete Head Branches:** Automatically delete the head branch after the pull request is merged.

1. **Archives**

* **Include Git LFS Objects:** Option to include Git Large File Storage objects in archives of the repository.

1. **Pushes**

* **Limit Branch Updates:** Restrict the number of branches or tags that can be updated in a single push (Beta).

1. **Danger Zone**

* **Change Repository Visibility:** Option to change the visibility of the repository (public, private).
* **Disable Branch Protection:** Option to disable branch protection rules.
* **Transfer Ownership:** Transfer the repository to another GitHub user or organization.
* **Archive Repository:** Archive the repository to make it read-only.
* **Delete Repository:** Permanently delete the repository.

1. **Collaborators**
2. **Moderation Options:** Set interaction limits for collaborators, including code review restrictions.
3. **Interaction Limits**
4. **Code Review Limits:** Restrict the interaction and review capabilities for specific users or groups.
5. **Branches**
6. **Branch Protection Rules:** Set rules for protecting branches.

* **Add Ruleset:** Create a new branch ruleset with specific protections.
* **Add Classic Branch Protection Rule:** Configure classic branch protection rules.

1. **Tags**
2. **Protected Tags:** Set rules to protect tags from being deleted or overwritten.
3. **Rules**
4. **Rulesets:** Define rulesets for the repository that apply to branches and tags.
5. **Actions**
6. **General:** Configure GitHub Actions for the repository.
7. **Runners:** Manage the runners available for executing GitHub Actions.
8. **Webhook**
9. **Add Webhook:** Set up webhooks to send repository events to external services.
10. **Environments**
11. **New Environment:** Create and configure environments for different stages of the development lifecycle.
12. **Codespaces**
13. **Prebuild Configuration:** Set up prebuild configurations for GitHub Codespaces.
14. **Set Up Prebuild:** Enable prebuild environments to speed up Codespace creation.
15. **Pages**
16. **Build and Deployment:** Configure GitHub Pages build settings.

* **Source:** Set the branch from which GitHub Pages should be deployed.
* **Deploy from Branch:** Specify the branch for GitHub Pages deployment.
* **GitHub Action:** Deploy GitHub Pages using a GitHub Action workflow.

1. **Branch:** Choose the branch for deployment.

* **Save:** Save the GitHub Pages settings.

1. **Visibility:** Control the visibility of GitHub Pages (public, private).

* **Try GitHub Enterprise**

1. **Code Security and Analysis:** Access advanced security and analysis features.
2. **Private Vulnerability Reporting**: Enable private reporting of vulnerabilities in the repository.
3. **Dependency Graph**

* **Automatic Submission:** Automatically submit dependencies for security analysis.

1. **Dependabot**

* **Dependabot Alerts:** Get notified of vulnerabilities in the repository’s dependencies.
* **Dependabot Security Updates:** Enable automatic updates for vulnerable dependencies.
* **Grouped Security Updates:** Bundle multiple security updates into a single pull request.
* **Dependabot Version Updates:** Enable Dependabot to update dependencies to newer versions.
* **Dependabot on Actions Runners:** Configure Dependabot for repositories using GitHub Actions.

1. **Code Scanning**

* **Tools:** Set up and configure code scanning tools like CodeQL.
* **CodeQL Analysis:** Enable and manage CodeQL analysis for the repository.
* **Other Tools:** Add additional code scanning tools.
* **Protection Rules:** Set rules for how code scanning alerts are handled.
* **Check Run Failure Threshold:** Define the failure threshold for code scanning check runs.
* **Security Alert Severity Level:** Set the severity level for security alerts.

1. **Secret Scanning:** Scan the repository for exposed secrets.
2. **Push Protection:** Protect against pushing sensitive information to the repository.
3. **Deploy Keys**
4. **Add Deploy Keys:** Add deploy keys to allow secure access to the repository.
5. **Secrets and Variables**
6. **Actions:** Manage secrets and variables for GitHub Actions workflows.
7. **Codespaces:** Manage secrets and variables for GitHub Codespaces.
8. **Dependabot:** Manage secrets and variables for Dependabot operations.
9. **GitHub Apps**
10. **Installed GitHub Apps:** View and manage GitHub Apps installed on the repository.

* **SonarCloud:** Integrate and configure SonarCloud for code analysis.

1. **Email Notifications**
2. **Configure:** Set up email notifications for repository activities.
3. **Pin**
4. **Pin Options:** Select which repositories to showcase on your profile.
5. **Rearrange Pins:** Drag and drop pinned repositories to change their order.
6. **Unwatch**
7. **Notification Settings:** Configure or customize how you are notified about repository activity.
8. **Fork**
9. **Fork Options:** Clone a repository to your account while maintaining a connection to the original project for pull requests.
10. **Star**
11. **Star List:** Access all repositories you've starred for easy reference.
12. **Branches**
13. **Create Branch:** Set up new branches to isolate work from the main codebase.
14. **Branch Protection:** Set rules to protect important branches (e.g., requiring pull requests for merging).
15. **Tags**
16. **Create Tag:** Add tags to specific commits to mark releases or significant updates.
17. **Tag Management:** View and manage existing tags in the repository.
18. **Go to File**
19. **Search Bar:** Type the name of the file to jump directly to it in the codebase.
20. **Add File**
21. **Create File:** Create a new file directly within the GitHub interface.
22. **Upload Files:** Drag-and-drop or select files to upload them to the repository.
23. **Create New File**
24. **File Editor:** Write and save code or text directly to the repository.
25. **Upload Files**
26. **Drag and Drop:** Drag files into the GitHub interface to add them to the codebase.
27. **Code**
28. **HTTPS**

* **Clone URL:** Copy the HTTPS URL to clone the repository with Git commands.

1. **SSH**

* **SSH Setup:** Requires SSH key setup on GitHub and your local machine.

1. **GitHub CLI**

* **CLI Usage:** Run GitHub-specific commands to manage repositories, issues, pull requests, and more.

1. **Clone Using the Web URL**

* **Copy URL:** Get the URL to use with Git or other version control tools.

1. **Open with GitHub Desktop**

* **Launch GitHub Desktop:** Click to open and manage the repository through the desktop application.

1. **Download Zip**

* **Offline Access:** Provides the codebase in a compressed file for offline use.

1. **About**
2. **Readme:** A section where the repository's README file is displayed for project information.
3. **Releases**
4. **Create Release:** Package and release versions of your software with notes and downloadable assets.
5. **Release Notes:** Provide detailed release notes for new features, bug fixes, and improvements.
6. **Packages**
7. **GitHub Packages:** View and publish packages such as Docker images, Node modules, or Ruby gems.
8. **Package Versions:** Access different versions of your packages directly from the repository.

## Creating a New Repository

1. Go to Repositories.
2. Click New.
3. Name the repository, choose between public/private.
4. Optionally add a README file.
5. Click Create Repository.

## Import Repository

1. Navigate to Repositories.
2. Click Import a Repository.
3. Paste the URL of the repository to import.
4. Click Begin Import.

## Your Profile

### Definition:

A user’s public-facing page that showcases their contributions, repositories, and activity on GitHub.

### Features:

1. Customizable bio and profile picture.
2. Displays repositories, stars, and contributions.
3. Public activity feed.
4. Followers and following lists.

### Advantages:

1. Highlights your open-source contributions.
2. Allows networking and collaboration.
3. Acts as a personal portfolio for developers.

### Disadvantages:

1. Limited customization options beyond basic profile information.
2. Activity visibility might raise privacy concerns.

### Types:

1. **Personal profile:** A user’s public profile on GitHub, showcasing their contributions and repositories.
2. **Organization profile:** A public profile for organizations, displaying their repositories and teams.

## Set Status

### Definition:

A feature allowing users to set a custom status visible to their team members on GitHub, indicating their current availability or focus.

### Features:

1. Customizable status message.
2. Option to include emojis.
3. Automatic expiration of status.
4. Option to limit visibility to specific teams or organizations.

### Advantages:

1. Clearly communicate availability.
2. Improve collaboration within teams.
3. Focus management by showing if you’re busy.

### Disadvantages:

1. Limited visibility to only users within your team or organization.
2. Can be overlooked if not frequently updated.

## GitHub Issues

### Definition:

GitHub Issues is a project management tool used for tracking bugs, feature requests, and other tasks within a repository.

### Features:

1. Create, assign, and comment on issues.
2. Link issues to pull requests.
3. Labels for categorizing issues.
4. Milestones for tracking progress.

### Advantages:

1. Simple and effective way to track tasks.
2. Seamlessly integrated with repositories and pull requests.
3. Supports collaboration with team members.

### Disadvantages:

1. Can become cluttered with too many issues.
2. Limited project management capabilities compared to dedicated tools.

### Types:

1. **Open Issues:** Active issues that have not yet been resolved or closed in a repository.
2. **Closed Issues:** Issues that have been resolved or marked as completed.

## Link a PR to an Issue

1. In a pull request, write Closes #[issue number] in the description.
2. Submit the PR to automatically link it to the issue.

## Create an Issue

1. Go to the Issues tab in the repository.
2. Click New Issue.
3. Add a title, description, and assign labels.
4. Click Submit.

## Create a Branch from an Issue

1. On the issue page, click Create Branch.
2. Name the branch and click Create.

## Assign Issues

1. Go to the issue.
2. Click the Assignees option.
3. Select the team member to assign the issue to.

## Add Assignees to Issues and Pull Requests

1. On an issue or PR, click Assignees in the sidebar.
2. Select the user(s) to assign.

## GitHub Pull Requests

### Definition:

GitHub Pull Requests allow developers to propose changes to a repository's codebase, which can be reviewed and merged by collaborators.

### Features:

1. Code review system with comments.
2. Ability to merge or reject proposed changes.
3. Visual comparison of changes (diffs).
4. Supports multiple merge methods (merge, squash, rebase).

### Advantages:

1. Simplifies collaboration on code changes.
2. Encourages peer review and improves code quality.
3. Tracks the entire development history of a feature or fix.

### Disadvantages:

1. Conflicts can arise during the merge process.
2. Complex workflows may require additional management.

### Types:

1. **Open Pull Requests:** Proposed changes to a codebase that are under review and not yet merged.
2. **Merged Pull Requests:** Pull requests that have been approved and integrated into the main codebase.
3. **Closed Pull Requests:** Pull requests that were either completed, rejected, or closed without merging.

### Status:

1. **Open:** This is the default status for a pull request. It indicates that the pull request is waiting for review.
2. **Closed:** The pull request has been merged or declined. The pull request is no longer active.
3. **Merged:** The pull request has been successfully integrated into the main branch.
4. **Draft:** A pull request is considered a draft when it's a work in progress and not yet ready for review.

### Draft PR:

1. **Work in progress:** Draft PRs signify that the changes are not yet ready for review and are still under development.
2. **Open for Feedback:** Once ready, the draft PR can be converted to a regular PR, opening it up for review and discussion.
3. **Ready to Merge:** After review and discussion, the PR can be merged into the target branch once all necessary changes have been addressed.

## Creating a New Pull Request

1. Navigate to the Pull Requests tab.
2. Click New Pull Request.
3. Choose the base and compare branches.
4. Enter a title and description.
5. Click Create Pull Request

## Commits in a Pull Request

1. Go to the Pull Request tab.
2. Review the commit history at the bottom of the PR page.

## Draft Pull Requests

1. When creating a PR, select Create Draft.
2. This marks the PR as "work in progress."

## Linking Activity in a Pull Request

1. Mention the issue or other PR by typing #issue or #PR.
2. GitHub will automatically create a link between them.

## GitHub Actions

### Definition:

GitHub Actions is a CI/CD (Continuous Integration/Continuous Deployment) tool that automates workflows, testing, and deployment directly from your repository.

### Features:

1. Predefined workflows for CI/CD automation.
2. Integration with third-party services and APIs.
3. Support for running jobs on GitHub-hosted or self-hosted runners.
4. Marketplace for reusable workflow templates.

### Advantages:

1. Seamless integration with GitHub repositories.
2. Automates tasks like testing, deployment, and build processes.
3. Reduces manual work and errors in deployment pipelines.

### Disadvantages:

1. Complex workflows may require significant setup.
2. Workflow execution may consume GitHub Actions minutes (paid).

### Types:

1. **Workflow templates:** Predefined workflows to automate development, testing, and deployment processes.
2. **GitHub-hosted runners:** Virtual machines provided by GitHub to run automated workflows.
3. **Self-hosted runners:** Custom servers configured by users to run their GitHub Actions workflows.

## GitHub Wiki

### Definition:

GitHub Wiki provides a space within each repository to document important information, such as how to use the project, project guidelines, and technical details.

### Features:

1. Built-in version control for documentation.
2. Markdown support for easy formatting.
3. Separate from the codebase for clean organization.

### Advantages:

1. Provides detailed project documentation in a centralized location.
2. Versioned history of wiki changes, just like code.
3. Accessible to all contributors.

### Disadvantages:

1. Not suitable for large or complex documentation projects.
2. Lacks advanced search and organizational features found in dedicated documentation tools.

### Types:

1. **Public wiki:** A publicly accessible wiki within a repository for documentation and collaborative notes.
2. **Private wiki (for private repositories):** A repository wiki that is only accessible to authorized users.

## Creating Wiki Pages

1. Navigate to the repository's Wiki tab.
2. Click New Page.
3. Add the title and content.
4. Click Save Page.

## Editing Wiki Pages

1. Open the Wiki.
2. Select the page and click Edit.
3. Make changes and click Save Page.

## Deleting Wiki Pages

1. Go to the Wiki tab.
2. Select the page and click Delete.

## GitHub Insights

### Definition:

GitHub Insights provides detailed metrics and analytics on repository activity, contributor behavior, and project health.

### Features:

1. Graphs showing commit history, code frequency, and contributor activity.
2. Issue and pull request tracking.
3. Contributor graphs showing code ownership and changes.
4. Community health overview.

### Advantages:

1. Provides a comprehensive view of project activity and health.
2. Helps managers track progress and contributions.
3. Easy to identify bottlenecks or inactive areas.

### Disadvantages:

1. Advanced insights may require third-party tools for deeper analysis.
2. Can be overwhelming for smaller repositories with low activity.

### Types:

1. **Contributor graphs:** Visual representations of contributions made by users to a repository over time.
2. **Community health overview:** A summary of key metrics and best practices to help manage and support a healthy project community.
3. **Code frequency graphs:** Visual data showing the amount of code added or deleted in a repository over time.

## Viewing Repository Insights

1. Go to the repository.
2. Click Insights from the top menu.
3. View metrics like traffic, commits, contributors, and dependencies.

## GitHub Settings

### Definition:

GitHub Settings allows users and organizations to configure various aspects of their repositories, including permissions, integrations, and security settings.

### Features:

1. Manage repository permissions and access control.
2. Configure branch protection rules.
3. Enable or disable repository features (issues, wiki, pull requests).
4. Integrate third-party services and apps.

### Advantages:

1. Centralized management of repository settings.
2. Granular control over permissions and security.
3. Easily configure integrations with external services.

### Disadvantages:

1. Complex configurations for large teams or organizations.
2. Requires ongoing maintenance to ensure proper settings are applied.

## GitHub Package

### Definition:

GitHub Package is a platform for hosting and managing packages (like Docker, npm, Maven, NuGet, etc.) alongside source code.

### Features:

1. Supports multiple package types (npm, Docker, RubyGems, etc.).
2. Seamless integration with GitHub Actions for CI/CD.
3. Private and public repositories for managing packages.
4. Versioning and dependency management.

### Advantages:

1. Centralized source code and package management.
2. Secure and private storage for internal packages.
3. Easy integration with GitHub CI/CD pipelines.

### Disadvantages:

1. Limited storage on free tiers.
2. Not as feature-rich as dedicated package management systems.

### Types:

1. **Public packages:** Software packages that are accessible to anyone on GitHub Packages.
2. **Private packages:** Software packages restricted to specific users or teams on GitHub Packages.

## Publishing your package

1. Go to the Packages section.
2. Click Publish a Package and follow the instructions.

## Viewing your published package

1. Go to the Packages tab in your repository.
2. Click on your published package.

## Installing a published package

1. Go to the package page.
2. Follow the instructions to install it using the appropriate package manager.

## GitHub Codespaces

### Definition:

GitHub Codespaces is a cloud-based development environment that lets you code, build, test, and debug directly from your browser or Visual Studio Code.

### Features:

1. Pre-configured development environments.
2. Customizable with devcontainer.json.
3. Integration with GitHub repositories.
4. Supports Visual Studio Code extensions.

### Advantages:

1. No need to install development tools locally.
2. Instant access to development environments.
3. Simplifies onboarding for new contributors.
4. Scalable resources based on project needs.

### Disadvantages:

1. Limited free-tier usage.
2. Requires a stable internet connection.

### Types:

1. **Browser-based Codespaces:** GitHub Codespaces running directly in a web browser without needing local setup.
2. **VS Code-based Codespaces:** GitHub Codespaces accessed and developed using Visual Studio Code on the desktop.

## Creating a new codespace

1. Open the repository.
2. Click the Code button and select Codespaces.
3. Click Create New Codespace.

## Sharing a Deep Link to a GitHub Codespace

1. In your Codespace, click Share.
2. Copy the deep link and share it.

## Adding and configuring dev containers

1. In the Codespace, click Configure Dev Container from the VS Code sidebar.
2. Modify the devcontainer.json file to add custom configurations.

## GitHub Discussion

### Definition:

GitHub Discussions is a platform within a repository where community members and contributors can engage in long-form discussions about projects.

### Features:

1. Separate space for discussions outside of issues and pull requests.
2. Supports Q&A, feedback, and brainstorming.
3. Ability to pin, close, or convert discussions into issues.
4. Markdown support for formatting posts.

### Advantages:

1. Encourages community collaboration.
2. Keeps issues and pull requests focused on development.
3. Easy to organize topics and threads.

### Disadvantages:

1. Can become cluttered if not managed.
2. Limited moderation tools compared to standalone forums.

### Types:

1. **Open discussions:** Active conversations within a repository's Discussions tab that are not yet resolved or closed.
2. **Pinned discussions:** Important discussions highlighted and fixed at the top of the Discussions tab for visibility.
3. **Converted issues:** Discussions that have been turned into GitHub Issues for tracking and resolving.

## Enabling GitHub Discussions on your repository

1. Go to the Settings tab of your repository.
2. Under Features, enable Discussions.
3. Discussions will appear as a new tab.

## Enabling GitHub Discussions on your organization

1. Go to the organization Settings.
2. Enable Discussions under communication settings.

## Contributions to your discussions

1. In the Discussions tab, click New Discussion.
2. Add your topic and start contributing.

## Setting up community guidelines for contributors

1. In the repository Settings, navigate to Community.
2. Create a CONTRIBUTING.md file and define guidelines.

## Creating a new discussion

1. Go to the Discussions tab.
2. Click New Discussion.
3. Add a title and description, then click Create Discussion.

## Creating a new poll

1. In the Discussions tab, create a new discussion.
2. Add poll options using Markdown (e.g., checkboxes).

## GitHub Page

### Definition:

GitHub Pages is a feature that allows users to host static websites directly from a GitHub repository.

### Features:

1. Supports custom domain names.
2. Free hosting for personal, project, and organization sites.
3. Automatically builds websites using Jekyll.
4. Markdown and HTML support.

### Advantages:

1. Free and easy to deploy static websites.
2. Integrated with GitHub repositories for automated updates.
3. Supports custom themes and configurations.

### Disadvantages:

1. Limited to static content; no backend functionality.
2. Requires some familiarity with HTML/CSS for customization.

### Types:

1. **Personal sites:** GitHub Pages used to host personal websites for individual users.
2. **Project sites:** GitHub Pages hosting websites that document or showcase a specific project.
3. **Organization sites:** GitHub Pages used by organizations to host websites for their projects or teams.

## Creating website

1. In your repository, go to Settings.
2. Scroll to GitHub Pages and select a source for your site.
3. Choose a theme and click Publish.

## Changing the title and description

1. In Settings, go to the Options tab.
2. Modify the title and description fields under Repository details.

## GitHub Saved Replies

### Definition:

GitHub Saved Replies allows users to create and store predefined responses for issues and pull requests to quickly respond with common messages.

### Features:

1. Create and manage custom responses.
2. Insert saved replies directly into issue or pull request comments.
3. Time-saving for handling repetitive messages.
4. Editable at any time.

### Advantages:

1. Saves time when managing multiple issues and pull requests.
2. Ensures consistent communication.
3. Easily reusable across different repositories.

### Disadvantages:

1. Limited to text responses (no advanced formatting or attachments).
2. Can result in impersonal communication if overused.

### Types:

1. **Repository-specific saved replies:** Predefined responses saved for use in a particular repository's discussions or issues.
2. **Account-wide saved replies:** Predefined responses available across all repositories within a user's or organization's account.

## Create Saved Replies

1. Go to your profile settings.
2. Click on "Saved Replies."
3. Create a new reply and save it for future use.

## Edit Saved Replies

1. Navigate to "Saved Replies" in your settings.
2. Click the edit icon next to the reply you want to update.

## Delete Saved Replies

1. In "Saved Replies," click the delete icon next to the reply you want to remove.

## GitHub Commit

### Definition:

A GitHub Commit refers to saving changes to a repository's codebase, creating a snapshot of the current state of the files and tracking changes over time.

### Features:

1. Tracks changes to a repository with unique commit IDs (hash).
2. Can include detailed commit messages explaining changes.
3. Supports multi-file and multi-line edits.
4. Links commits to issues and pull requests.

### Advantages:

1. Detailed version control with a complete history of changes.
2. Helps collaborate by tracking contributions.
3. Easy to revert changes to a previous state.

### Disadvantages:

1. Poor commit messages can make it hard to track changes.
2. Large commits make it harder to isolate bugs.

### Types:

1. **Regular commits:** Standard changes made to a codebase, tracking incremental development.
2. **Merge commits:** Commits generated when integrating one branch into another, preserving the history of both.
3. **Squash commits:** Single commits created by combining multiple changes, often used to streamline the commit history.

## Commenting on Code

1. Go to the file in the PR.
2. Hover over the code line and click Add Comment.

## Marking a Comment as an Answer

1. In a discussion or issue, click Mark as Answer for a comment.

## Commit your first change

1. Modify a file in the repository.
2. Stage the changes using git add.
3. Commit the changes using git commit -m "message".
4. Push the commit with git push.

## Milestone

### Definition:

A GitHub Milestone is a tool used to group issues and pull requests by a common goal or release, allowing teams to track progress toward a specific objective.

### Features:

1. Group issues and pull requests into milestones.
2. Track progress via percentage of completed tasks.
3. Use milestones for release management or sprint planning.

### Advantages:

1. Visualizes progress toward specific goals.
2. Helps manage releases or project phases.
3. Supports agile project management workflows.

### Disadvantages:

1. Limited functionality for complex project management.
2. Manual updating may be required to track accurate progress.

### Types:

1. **Open Milestones:** Active milestones tracking the progress of issues or pull requests towards a goal.
2. **Closed Milestones:** Milestones that have been completed or archived.

## Label

### Definition:

Labels are a way to categorize issues and pull requests in GitHub repositories, helping to organize and prioritize tasks.

### Features:

1. Color-coded labels for quick identification.
2. Customizable label names and categories.
3. Search and filter issues based on labels.
4. Can be applied to issues and pull requests.

### Advantages:

1. Helps prioritize and organize tasks.
2. Makes it easy to filter and manage issues.
3. Supports collaboration by defining task categories.

### Disadvantages:

1. Overuse of labels can cause confusion.
2. Needs consistent management for effective use.

### Types:

1. **Default labels (bug, enhancement):** Predefined tags for categorizing issues or pull requests, such as "bug" or "enhancement."
2. **Custom labels (urgent, needs review):** User-defined tags for organizing and prioritizing issues or pull requests, such as "urgent" or "needs review."

## Announcement

### Definition:

Announcements in GitHub are public or private communications within a project or organization, often used to inform contributors of important updates.

### Features:

1. Provides important project updates or changes.
2. Can be pinned or highlighted in discussions or repositories.
3. Supports Markdown for rich text formatting.

### Advantages:

1. Centralized communication for teams and communities.
2. Keeps contributors informed of major changes or milestones.

### Disadvantages:

1. Can be overlooked if contributors are not actively monitoring the repository.
2. Limited interaction options compared to issues or discussions.

### Types:

1. **Public announcements:** Notices shared with all users or the public regarding important updates or events.
2. **Private or team-specific announcements:** Notices shared within a specific team or private group for internal communication.

## Ideas

***Definition:***  
GitHub Ideas is a way for contributors to suggest improvements, features, or solutions in a repository, often through Discussions.

### Features:

1. Community-driven feedback and brainstorming.
2. Integrated with issues for easy tracking of ideas.
3. Can be voted and commented on by others.

### Advantages:

1. Encourages collaboration and new feature development.
2. Provides insight into community needs and desires.

### Disadvantages:

1. Unmoderated ideas can lead to irrelevant suggestions.
2. May require significant review and prioritization.

### Types:

1. **Feature ideas:** Suggestions for new features or enhancements for a project.
2. **Improvement suggestions:** Recommendations for refining or optimizing existing features or processes.

## Polls

***Definition:***  
Polls in GitHub are often used in discussions or issues to gather feedback or opinions from contributors.

### Features:

1. Quick and easy way to gather votes.
2. Can be used to decide on features or next steps in a project.
3. Results are visible in real-time.

### Advantages:

1. Facilitates decision-making by collecting majority opinions.
2. Reduces lengthy discussions by simplifying choices.

### Disadvantages:

1. Polls may oversimplify complex decisions.
2. Results may not always reflect informed opinions.

### Types:

1. **Single-choice polls:** Surveys allowing participants to select one option from multiple choices.
2. **Multiple-choice polls:** Surveys allowing participants to select more than one option from a list.

## Code Review

***Definition:***  
Code Review is the process of examining code changes in pull requests to ensure quality, consistency, and security before merging into the main branch.

### Features:

1. Inline commenting and suggestions on pull request code.
2. Supports multiple reviewers with approval requirements.
3. Integrates with CI/CD workflows.

### Advantages:

1. Improves code quality and catches bugs early.
2. Promotes collaboration and knowledge sharing.

### Disadvantages:

1. Can slow down development if not well-managed.
2. May lead to bottlenecks in large teams.

### Types:

1. **Peer reviews:** Evaluation of code or contributions by other developers to ensure quality and compliance.
2. **Automated code reviews (using tools like GitHub Actions):** Code quality checks performed automatically using integrated tools.

## Achievements

***Definition:***  
GitHub Achievements are badges or milestones that recognize contributions and engagement on the platform.

### Features:

1. Recognizes contributors' efforts in open-source projects.
2. Displays badges on GitHub profiles.
3. Milestones for activities like creating repositories, opening pull requests, and contributions to discussions.

### Advantages:

1. Encourages active participation.
2. Highlights contributor expertise and experience.

### Disadvantages:

1. May lead to badge hunting, prioritizing quantity over quality.
2. Limited visibility of achievements outside GitHub.

### Types:

1. **Contributor badges:** Recognition icons awarded to users for their contributions to a project.
2. **Maintainer badges:** Recognition icons awarded to users who manage or maintain a project.

## github.dev

***Definition:***  
github.dev is a web-based code editor that allows you to view and edit your GitHub repositories directly in the browser without needing a local environment.

### Features:

1. Fast, lightweight web editor.
2. Supports Visual Studio Code extensions.
3. Direct integration with GitHub repositories for real-time editing.

### Advantages:

1. No setup required; instant access to repositories.
2. Supports quick fixes and reviews directly in the browser.

### Disadvantages:

1. Limited features compared to full-fledged IDEs.
2. Requires a stable internet connection.

### Types:

1. **Browser-based code editing:** Editing code directly within a web browser interface without needing local tools.

## GitHub Markdown

***Definition:***GitHub Markdown is a lightweight markup language used to format text in GitHub, supporting rich text editing in issues, pull requests, and more.

### Features:

1. Supports headings, lists, links, images, code blocks, and more.
2. Integrated with GitHub Flavored Markdown (GFM) for additional syntax.
3. Used across discussions, issues, README files, etc.

### Advantages:

1. Simple and easy to learn.
2. Enhances communication by providing structured text formatting.

### Disadvantages:

1. Limited advanced formatting compared to full document editors.
2. Requires knowledge of Markdown syntax.

### Types:

1. **GitHub Flavored Markdown (GFM):** A variant of Markdown with GitHub-specific extensions for enhanced formatting.
2. **Basic Markdown:** The standard version of Markdown for simple text formatting.

## Slash Commands

***Definition:***  
Slash Commands are quick shortcuts typed directly into GitHub comments or pull requests to perform actions like closing an issue or assigning a reviewer.

### Features:

1. Commands like /close, /assign, /label.
2. Speeds up common tasks without leaving the comment thread.
3. Integrated into the GitHub platform.

### Advantages:

1. Saves time by eliminating the need for multiple clicks.
2. Streamlines issue and pull request management.

### Disadvantages:

1. Limited range of available commands.
2. Requires familiarity with the available commands.

### Types:

1. **Issue management commands:** Commands used to manage and organize issues within a repository.
2. **Pull request commands:** Commands used to handle and process pull requests.

## GitHub Desktop

***Definition:***  
GitHub Desktop is a GUI application that simplifies the use of Git and GitHub, allowing users to manage repositories without using the command line.

### Features:

1. Visual interface for managing branches, commits, and pull requests.
2. Seamless synchronization with GitHub repositories.
3. Supports Git features like merge conflicts, diffs, and more.

### Advantages:

1. Easy to use for beginners.
2. Streamlines workflows by eliminating the need for command-line knowledge.

### Disadvantages:

1. Limited advanced Git operations compared to the CLI.
2. Slower than terminal-based operations for experienced users.

### Types:

1. **macOS version:** The version of GitHub's software or tools compatible with macOS.
2. **Windows version:** The version of GitHub's software or tools compatible with Windows.

## GitHub Mobile

***Definition:***  
GitHub Mobile is the mobile application that allows you to manage your GitHub repositories, issues, and pull requests on the go.

### Features:

1. Review and merge pull requests.
2. Respond to issues and discussions.
3. Manage notifications and repositories.

### Advantages:

1. Provides flexibility by allowing work from anywhere.
2. Ideal for quick reviews or updates.

### Disadvantages:

1. Limited functionality compared to the desktop or browser versions.
2. Not ideal for extensive coding or project management tasks.

### Types:

1. **iOS app:** The GitHub application available for iOS devices.
2. **Android app:** The GitHub application available for Android devices.

## GitHub Administration

***Definition:***  
GitHub Administration refers to the management and configuration of GitHub organizations, repositories, and user permissions.

### Features:

1. Controls access and permissions for repositories.
2. Manages billing, security, and organization settings.
3. Allows repository ownership and role management.

### Advantages:

1. Centralized control over large projects or organizations.
2. Enhances security and permission management.

### Disadvantages:

1. Requires careful configuration to avoid permission issues.
2. Can be complex for large teams.

### Types:

1. **Repository admin:** A user role with full administrative rights over a repository's settings and management.
2. **Organization admin:** A user role with full administrative rights over an organization's repositories and settings.

## Text Formatting Toolbar

***Definition:***  
The Text Formatting Toolbar in GitHub provides an interface to apply Markdown formatting without manually typing syntax.

### Features:

1. Bold, italic, headers, lists, links, and more.
2. Live preview of formatting.
3. Available in issues, pull requests, and discussions.

### Advantages:

1. Simplifies Markdown usage for beginners.
2. Makes formatting faster and more accessible.

### Disadvantages:

1. Limited to common formatting options.
2. Does not support custom formatting beyond Markdown.

### Types:

1. **Standard formatting toolbar (Markdown):** A toolbar providing basic formatting options for Markdown text.
2. **GitHub Flavored Markdown (GFM) toolbar:** A toolbar with advanced formatting options specific to GitHub Flavored Markdown.

## Open Source

***Definition:***  
Open Source refers to publicly available software projects on GitHub where anyone can view, modify, and contribute to the source code.

### Features:

1. Public repositories for collaborative development.
2. Free for anyone to contribute.
3. Version control and transparent development.

### Advantages:

1. Encourages collaboration and innovation.
2. Free access to high-quality code and projects.

### Disadvantages:

1. Potential for unvetted contributions.
2. Limited control over contributions.

### Types:

1. **Public open-source projects:** Projects whose source code is freely available and can be modified by anyone.
2. **Licensed open-source projects (MIT, GPL, etc.):** Open-source projects distributed under specific licenses like MIT or GPL that dictate usage and modification rights.

## InnerSource

***Definition:***  
InnerSource is the practice of applying open-source methodologies within a private organization, allowing internal teams to collaborate on shared code.

### Features:

1. Private repositories shared across internal teams.
2. Open contribution model within an organization.
3. Encourages reuse and collaboration on internal projects.

### Advantages:

1. Encourages cross-team collaboration.
2. Increases code reuse and reduces duplication.

### Disadvantages:

1. Requires strong governance to ensure quality.
2. Can lead to ownership conflicts over shared code.

### Types:

1. **Internal repositories (private):** Repositories restricted to specific users or teams within an organization.
2. **Cross-functional teams contributing to shared code:** Diverse teams from different functions working together on a shared codebase.

## EMUs (Enterprise Managed Users)

***Definition:***  
EMUs are GitHub Enterprise Managed Users, allowing organizations to manage their users centrally via an identity provider like Active Directory.

### Features:

1. Centralized control over user accounts.
2. Integrated with enterprise identity management systems.
3. Provides audit and compliance tools for enterprise environments.

### Advantages:

1. Enhances security and compliance for large organizations.
2. Streamlines user management with Single Sign-On (SSO).

### Disadvantages:

1. Requires integration with identity providers.
2. Higher cost for enterprise plans.

### Types:

1. **Managed users (enterprise only):** Users whose access and roles are controlled centrally in an enterprise environment.
2. **Federated user accounts:** User accounts that are linked or synchronized across different systems or organizations.

## GitHub Marketplace

***Definition:***  
GitHub Marketplace is a platform for discovering and integrating third-party tools, apps, and services into your GitHub workflow.

### Features:

1. Thousands of applications for CI/CD, security, automation, and more.
2. One-click installation and configuration for GitHub repositories.
3. Supports both free and paid apps.

### Advantages:

1. Simplifies the process of integrating tools into GitHub workflows.
2. Provides access to a large ecosystem of third-party tools.

### Disadvantages:

1. Paid apps may require additional costs.
2. Quality of tools may vary.

### Types:

1. **Free applications:** Software applications available at no cost to users.
2. **Paid/proprietary applications:** Software applications that require a purchase or subscription and are owned by a specific entity.