UNIT 5

Unit 5

The GitHub EcoSystem: Exploring the GitHub Marketplace - Introducing the GitHub Marketplace, Listing Your App on the Marketplace, Considering Common Apps to Install. **GitHub and You -** Understanding Your GitHub Profile, Starring Repositories.. **The Parts of TENS : Ten WaAttending Events-** Exploring Types of Events, Knowing What to Expect at Events, Becoming Familiar with GitHub Events, Speaking at Event**sys to Level Up on GitHub -** GitHub Help Docs, GitHub Learning Labs, GitHub In-Person Training, Project-Specific Documentation, External Community Places, Online Coding Tutorials, Online Courses and Tutorials, Blogs and Twitter, Community Forum. **Ten Ways to Improve Your Development Workflow -** Drafting Pull Requests, Git Aliases, Run Tests Automatically, Take Breaks, Prototype User Interfaces, Scaffold Apps with Yeoman, Chrome Web Developer Tools, StackOverflow, Code Analysis Tools, Project Boards.

The Marketplace vetting process

- The **GitHub Marketplace** is a platform provided by GitHub that offers a curated collection of third-party tools, applications, and services designed to integrate seamlessly with GitHub. It helps developers and teams enhance their workflows by providing easy access to tools for automation, collaboration, testing, security, and more.

The Marketplace vetting process

- The GitHub **Marketplace vetting process** ensures that applications listed in the Marketplace meet specific quality, security, and usability standards. This process provides users with confidence that the apps they install are safe, reliable, and valuable.

- Apps must meet the following four categories of requirements before being listed in the GitHub Marketplace:
- **a) User Experience**
- Apps must meet certain usability and behavior standards to provide a good user experience.
- Requirements include:
  - The app should already have a minimum number of users and installs.
  - The app must include clear documentation and links to help users understand how to use it.
  - The app should not persuade users to leave GitHub or use competing services.
  - The app must deliver genuine value to customers (e.g., no frivolous or non-useful apps like "Fart apps").

- **Brand and Listing**
- Focuses on branding and presentation:
  - Each app must have its own unique logo.
  - If GitHub's logo is used, it must comply with GitHub’s **Logos and Usages Guidelines**.
  - The listing must meet the branding and description standards outlined by GitHub, ensuring a professional and consistent representation of the app.

- **Security**
- GitHub conducts a **security review** of each app before it is listed.
  - The review ensures that the app adheres to security best practices, protecting user data and preventing vulnerabilities.
  - Developers are provided with detailed security guidelines in a document available on GitHub's developer platform.

- **Billing Flows**
- Every Marketplace app must integrate with GitHub's **billing system**:
  - Users can subscribe to the app and cancel their subscription using the payment details already on file with GitHub.
  - Any changes made via GitHub (e.g., subscription modifications) must be immediately reflected in the app’s website or service.
  - This integration streamlines the billing process for users and ensures consistency between GitHub and the app.

Listing Your App on the GitHub Marketplace

- The GitHub Marketplace provides an opportunity to showcase your app to a larger audience of developers and organizations. However, listing your app involves a structured process to ensure it meets GitHub’s quality and security standards.

- **Steps to List Your App on the GitHub Marketplace**

1.Start the Submission Process:

1. Navigate to the GitHub Marketplace submission page:
    1. Click the **Submit Your Tool for Review** link at the bottom of the Marketplace landing page.
    2. Alternatively, visit https://github.com/marketplace/new.
2. This page will display your existing applications eligible for listing.

2.Create a Draft Listing:

1. Select the app you want to list by clicking the **Create Draft Listing** button.
2. Enter the **name of the listing** and choose an appropriate category for your app.

3.Save and Edit the Draft:

1. You can save the draft and revisit it anytime via https://github.com/marketplace/manage.

1.Fill in Listing Details: Complete the following steps to create a comprehensive listing:

2.a) Add Contact Information:

1. Provide three email addresses for:
    1. **Technical Lead**
    2. **Marketing Lead**
    3. **Finance Lead**

3.b) Listing Description:

1. Add a detailed product description.
2. Include a **logo** and **screenshots** to visually represent your app.
3. This information will appear on your app's Marketplace page.

4.c) Set Up Plans and Pricing:

1. Define the pricing structure:
    1. Free plans.
    2. Monthly subscriptions.
    3. Monthly per-user plans.
    4. Optional: Add a **14-day free trial**.

5.d) Set Up a Webhook:

1. Provide a **URL for webhook events**, which will notify you of:
    1. Purchases
    2. Cancellations
    3. Changes like upgrades and downgrades

6.e) Accept the Marketplace Developer Agreement:

1. Review and agree to the **Marketplace Developer Agreement**, which outlines the terms for listing your app.

7.Submit for Review:

1. Once all the details are completed, click the **Submit for Review** button.
2. GitHub employees will review your submission to ensure it meets the requirements.
3. You will receive an email notification about the outcome of the review.

Considering Common Apps to Install

- Continuous integration (CI) apps automatically build and test your code every time you push it to GitHub. If you have a CI app, such as AppVeyor, installed on your repository, you see the status of the check at the bottom of each pull request, as shown in Figure 15-5.

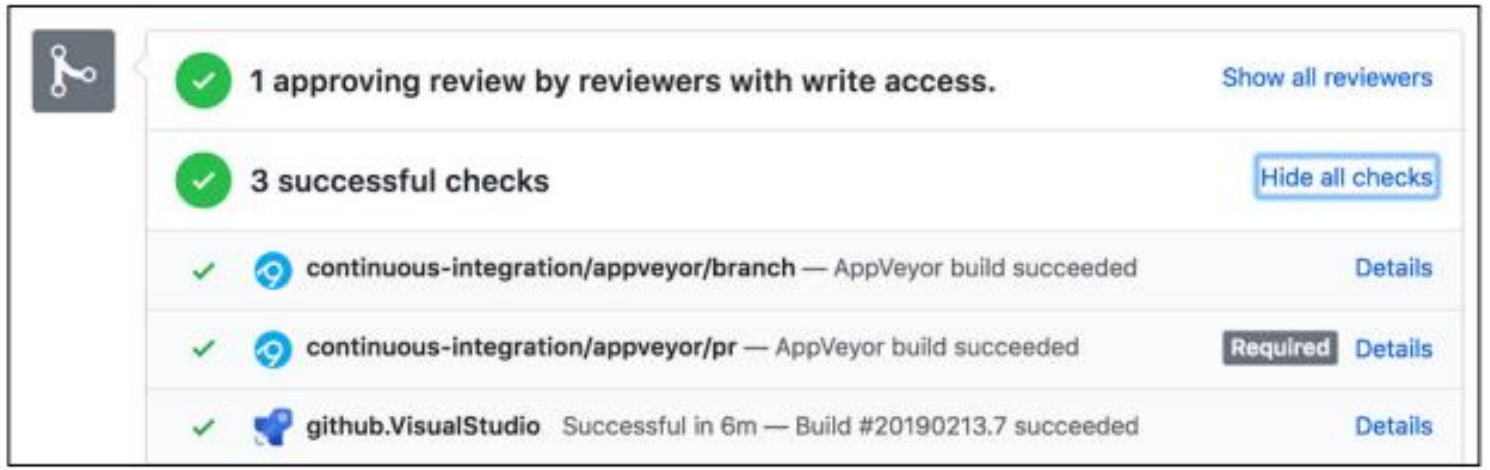

FIGURE 15-5: AppVeyor CI app example on the GitHub for Visual Studio repository.

If you're the owner of the repository, you can also specify whether checks have to pass before the branch can be merged into the main branch. Just head into the Settings tab. If you have any rules on the main branch already, click Edit; otherwise, click Add Rule. From there, you can scroll down and select Require Status Checks to Pass before merging

Code Quality

- **Purpose**: Ensures code follows style, quality, security, and test coverage standards.
- **Benefits**:
  - Helps maintain consistent styling and quality in the codebase.
  - Reduces the likelihood of bugs by enforcing coding standards.
  - Ensures good test coverage to identify untested areas.
- **Examples**:
  - **Rubocop**: Checks the style of Ruby code and provides feedback on coding practices.
  - **Codecov**: Tracks test coverage and comments on pull requests with the percentage of code covered by tests.

Localization

•**Purpose**: Facilitates publishing apps in multiple languages.
•**Benefits**:
•Simplifies the process of translating app content, such as documentation or UI text.
•Enables global collaboration for translation tasks.
•Automates pull requests for translation updates when accuracy thresholds are met.
•**Example**:
•**Crowdin**: Links repositories to Crowdin accounts, allowing contributors worldwide to help translate content.
• Open-source projects can use Crowdin for free.

Monitoring

- **Purpose**: Tracks performance, errors, and dependencies.
- **Benefits**:
  - Provides real-time notifications for dependency updates.
  - Monitors code for performance issues and potential errors.
- **Examples**:
  - **Greenkeeper**: Notifies and updates JavaScript dependencies by opening pull requests.
  - **Other tools**: Measure performance metrics and monitor error logs.

4. Dependency Management

- **Purpose**: Manages external libraries and dependencies used in a project.
- **Benefits**:
- Ensures dependencies are up-to-date.
- Allows control over public and private dependencies.
- **Examples**:
- **Dependabot**: Automatically checks and updates dependencies by submitting pull requests.
- **MyGet**: Provides a private package registry for enterprises, allowing internal packages to remain secure.

- **Testing**
- **Purpose**: Streamlines testing tasks as part of the software development lifecycle.
- **Benefits**:
  - Helps create, run, and monitor test plans and test runs.
  - Enhances quality assurance processes by integrating testing directly into GitHub.
- **Example**:
  - **TestQuality**: Integrates with GitHub to assist in coordinating and tracking software testing tasks.

- **Learning**
- **Purpose**: Provides interactive lessons for learning GitHub.
- **Benefits**:
  - Walks users through tasks to build GitHub skills.
  - Offers a self-paced learning experience.
- **Example**:
  - **GitHub Learning Lab**: Installs a bot that provides step-by-step guidance on GitHub usage through practical tasks.

Attending events

- Attending events is a powerful way for software developers to grow their careers, build connections, and learn new skills.
- **Why Attend Events?**

1. **Career Growth**:
    1. Meet experts, mentors, and peers to expand your professional network.
    2. Learn about new technologies, tools, and methodologies.
    3. Speaking at events enhances visibility and credibility in the developer community.
2. **Types of Events**:
    1. Ranges from informal meet-ups to international conferences.
    2. Great for exposure to diverse topics and professional opportunities.

- **Exploring Types of Events**

1. **Meet-ups and User Groups**:
    1. **Description**: Small, informal gatherings covering specific topics, often hosted monthly.
    2. **Purpose**: Dip your toes into developer events; connect with local peers.
    3. **Examples**:
        1. Brooklyn JS (Brooklyn, NY): brooklynjs.com
        2. .NET São Paulo (Brazil): meetup.com/dotnet-Sao-Paulo

- **Regional Conferences**:
- **Description**: Focused on local developers, often 1–2 days long, featuring talks and workshops.
- **Benefits**: Affordable, localized opportunities for skill-building and networking.
- **Examples**:
    - Caribbean Developers Conference: cdc.dev
    - JSConf Chile: jsconf.cl

- **Hackathons**:
- **Description**: Collaborative coding events where teams build solutions to specific problems.
- **Purpose**: Experiment, learn new technologies, and build prototypes.
- **Example**: Microsoft Imagine Cup imaginecup.microsoft.com

- **Major Conferences**:
- **Description**: Large-scale events attracting global audiences.
- **Features**: Multiple tracks, hands-on labs, and major product announcements.
- **Examples**:
  - GitHub Universe: githubuniverse.com
  - Apple WWDC: developer.apple.com/wwdc

- **GitHub-Specific Events**

1.GitHub Universe:

1. Flagship annual conference with major announcements and sessions.

2.GitHub Satellite:

1. Smaller, regional versions of Universe held worldwide.

3.GitHub Constellation:

1. Local community events focusing on specific developer communities.

4.Git Merge:

1. Git-focused conference with hands-on workshops.

- **Key Features of Events**

1. **Keynotes**:
    1. Set the tone for the event and often include major announcements.
2. **Session Tracks**:
    1. Organized talks focusing on themes, with time for Q&A.
3. **Hallway Tracks**:
    1. Informal, unscheduled interactions between attendees.
4. **After-Hour Events**:
    1. Social gatherings that provide opportunities for informal networking.

Ten Ways to Level Up on GitHub

- **Ten Ways to Level Up on GitHub**
- Mastering GitHub involves developing skills in using its features, contributing to software projects, and engaging with the community. Here’s a detailed breakdown of ten ways to improve your expertise:

- **Trial and Error**
- **How It Helps**: Experimentation is one of the best ways to learn GitHub. Create repositories, test features like pull requests and issues, and explore collaboration tools.
- **Tip**: Invite friends to collaborate on test repositories for hands-on experience. Practice using dummy repositories for new features.

- **GitHub Help Docs**
- **What It Offers**: Extensive documentation covering all GitHub features.
- **How to Use**:
  - Access at GitHub Help.
  - Use the search bar to find specific topics.
  - Follow cross-referenced links and the **Further Reading** section for additional resources.
- **Pro Tip**: If documentation doesn't resolve your issue, use the **Contact a Human** feature.

- **GitHub Skills**
- **Description**: Interactive, step-by-step tutorials to learn GitHub workflows.
- **Features**:
  - Courses on topics like merge conflict resolution, GitHub Actions, and open-source contributions.
  - Use the bot-guided experience to follow instructions and practice real scenarios.
- **Get Started**: Visit GitHub Skills to explore available courses.

- **GitHub In-Person Training**
- **Purpose**: Professional training sessions tailored to organizations.
- **Focus Areas**:
  - Workflow consultation, API integration, admin mentoring, and more.
- **Use Case**: Ideal for teams adopting GitHub as part of their workflow.
- **Learn More**: Visit GitHub Training Services.

- **Project-Specific Documentation**
- **Why It’s Useful**: Understand how to contribute to specific open-source projects.
- **How to Use**:
  - Read README files and contribution guides (e.g., VS Code’s How to Contribute).
  - Explore coding style guidelines for consistency.
  - Review open and closed pull requests to learn from community interactions.

- External Community PlatformsExamples:
- Stack Overflow: Search for questions tagged with specific technologies like visual-studio-code.
- Discord: Join servers for real-time discussions, like Sentry’s public Discord.
- GitHub Discussions: Use discussion forums for questions and general feedback on projects.Benefit: Provides a less formal way to learn and interact with the community.

- **Online Coding Tutorials**
- **Interactive Learning**:
  - Platforms like JSFiddle and Try .NET offer in-browser sandboxes.
  - Ruby learners can start with Try Ruby.
- **Tip**: Practice typing code manually for better retention.

- **Online Courses**
- **Popular Platforms**:
  - **Coursera**: Offers certifications like Ruby on Rails Web Development.
  - **EdX**: Provides professional certificates and MicroMasters programs.
  - **Udemy** and **Pluralsight**: Affordable and beginner-friendly courses.
  - **Khan Academy**: Free courses for all ages on various topics.
- **Use Case**: Ideal for deep dives into software development concepts and frameworks.

- **Blogs, YouTube, and Social Media**
- **Finding Resources**:
  - Follow GitHub's blog (GitHub Blog) for updates and tips.
  - Explore tutorials on **Dev.to** (Dev.to GitHub Topics).
  - Subscribe to YouTube channels like **LearnCode.academy** or **Siraj Raval** for tutorials and insights.
- **Tip**: Social media accounts like GitHub's Twitter (@GitHub) often highlight community contributions and events.

- **Community Forums**
- **GitHub Community Discussions**:
  - A dedicated space for questions about GitHub features and workflows.
  - Visit: GitHub Community Discussions.
- **GitHub Education Forum**:
  - Focused on using GitHub in academic settings (GitHub Education Forum).
- **Benefit**: Interact with a global network of developers to learn and share knowledge.

Ten ways to improve your development workflow

- Drafting Pull Requests
- Run Tests Automatically
- Take Breaks
- Prototype User Interfaces
- Chrome Web Developer Tools:: Debug and optimize web applications.
- StackOverflow: Resolve programming challenges.
- Code Analysis Tools
- Project Boards : Visualize and manage project tasks.