OPEN SOURCE ENGINEERING

Student ID: 2400032275

1 Understanding the Core Ubuntu Linux Distribution

1.0.1 1. Overview and Philosophy

Ubuntu is a powerful, free, and open-source operating system built upon the stable foundation of Debian Linux. It stands as the world's most popular Linux distribution for desktop use, successfully blending cutting-edge features with unparalleled user-friendliness. Developed and maintained by Canonical Ltd., Ubuntu's guiding principle is "Linux for human beings." This philosophy drives its commitment to accessibility, stability, and providing an intuitive computing experience for everyone, from novice users to seasoned developers.

1.0.2 2. The Desktop Experience (GNOME)

The standard Ubuntu desktop utilizes the **GNOME** desktop environment, which presents a modern, clean, and highly efficient graphical interface. The key design elements include a permanent dock (launcher) on the left side for quick access to essential applications, and the **Activities Overview**. This view, easily accessed by pressing the Super (Windows) key, provides a centralized hub for managing all open windows, workspaces, and system-wide searching. This streamlined workflow makes Ubuntu feel contemporary and ensures high productivity. Furthermore, Ubuntu is recognized for its strong, out-of-the-box hardware detection and compatibility, simplifying the setup process for most users.

1.0.3 3. Software Management and Packaging

Ubuntu employs a robust dual-system for software management. The traditional and reliable **Advanced Packaging Tool (APT)** manages **DEB** packages, handling core system utilities and standard applications sourced from official repositories. Complementing this is the use of **Snaps**, a modern, containerized package format pioneered by Canonical. Snaps bundle an application with all its required dependencies, guaranteeing consistent performance across different Ubuntu versions. Crucially, Snaps run in a **sandboxed** environment, isolating them from the rest of the operating system to significantly enhance overall application security. This flexibility ensures users have access to a vast, up-to-date, and secure software library.

2 Encryption and GPG

2.1 Types of Encryption in Ubuntu

Ubuntu offers two primary approaches to encryption: **Full Disk Encryption (FDE)** and **File/Directory Encryption**.

2.1.1 1. Full Disk Encryption (FDE)

- **What it is:** FDE encrypts the entire hard drive or a large partition, including the operating system files, swap space, and user directories.
- **How it works:** Ubuntu uses **LUKS** (Linux Unified Key Setup) for FDE. When the system boots, you are prompted for a **passphrase**. If correct, LUKS decrypts the entire drive, and the decryption process runs transparently in the background while the system is in use.

- **Purpose:** The primary defense against data loss due to **theft** or **physical access** to the computer when it is turned off. If someone steals the hard drive, the data is useless without the LUKS passphrase.
- **Implementation:** FDE is typically enabled during the Ubuntu installation process by selecting the "Encrypt the new Ubuntu installation" option. It's much more difficult to enable after installation.

2.1.2 2. File and Directory Encryption

- **What it is:** This method encrypts specific files, directories, or messages, offering granular control over which data is protected.
- **Tools:**
  - **GPG (GNU Privacy Guard):** The standard, used for encrypting individual files and especially for secure communication using **public-key cryptography**.
  - **eCryptfs (older):** Previously used for encrypting the user's Home directory, but has been largely phased out for FDE.

2.2 GPG (GNU Privacy Guard) Explained

GPG is the GNU implementation of the **OpenPGP** standard (originally Pretty Good Privacy - PGP). It is essential for protecting individual files and ensuring secure, authenticated communication.

2.2.1 1. Core GPG Concepts

GPG relies on **asymmetric cryptography**, which uses a pair of mathematically linked keys:

- **Public Key:** This key is shared with everyone. It can be used to **encrypt** a message that only you can read, or to **verify** a signature you created.
- **Private (Secret) Key:** This key is kept **secret** and is protected by a strong passphrase. It is used to **decrypt** messages sent to you, or to **digitally sign** files to prove they came from you.

2.2.2 2. Basic GPG Command-Line Usage

GPG is usually pre-installed on Ubuntu and is primarily used through the command line (Terminal).

A. Generating a Key Pair The first step is to create your public and private key pair:
Bash

```
gpg --full-generate-key
```

You will be prompted to select the key type (RSA and RSA is common), keysize (4096 is recommended), expiration date, and your Real Name, Email, and a strong **passphrase** to protect your private key.

B. Encrypting a File for Yourself (Symmetric Encryption) To quickly encrypt a file using a single passphrase (like a standard password), use symmetric encryption:

Bash

```
gpg -c myfile.txt
```

This command will prompt you for a passphrase and create an encrypted file named `myfile.txt.gpg`.

C. Encrypting a File for Someone Else (Asymmetric Encryption) To securely send a file, you must use the recipient's **Public Key** (which you must have previously imported into your keyring with `gpg --import`):

Bash

```
gpg --encrypt --recipient "recipient@example.com" mysecretfile.doc
```

This creates `mysecretfile.doc.gpg`. Only the recipient, who holds the corresponding Private Key, can decrypt it.

D. Decrypting a File To decrypt a file that was encrypted for you:

Bash

```
gpg --decrypt mysecretfile.doc.gpg
```

You will be prompted for the passphrase that protects your Private Key. You can use the `--output` option to specify the decrypted

3 Sending Encrypted Email

3.1 Prerequisite: Setting Up GPG

Before you can send or receive encrypted mail, both you and your recipient must have GPG keys set up and exchanged:

1. **Generate Keys:** Both parties must have generated a public/private key pair using GPG (as discussed previously, using `gpg --full-generate-key`).
2. **Exchange Public Keys:** You need the recipient's **Public Key**, and they need your Public Key. You can exchange these by:
   - **Exporting** the key: `gpg --armor --export 'Recipient Name' > recipient_key.asc` and sending the `.asc` file.
   - **Uploading** the key to a public key server.
3. **Import Key:** You must import the recipient's key into your GPG keyring: `gpg --import recipient_key.asc`.

3.2 Sending the Encrypted Email

The most common and user-friendly way to send GPG-encrypted emails on Ubuntu is by using **Mozilla Thunderbird** with the **Enigmail** add-on (or its built-in equivalent in modern versions of Thunderbird).

3.2.1 1. Compose the Message

- **Open Thunderbird** and start composing a new email.
- Write your message as usual.

3.2.2 2. Encryption and Signing

You will use the GPG function built into the mail client to perform two critical steps:

1. **Encryption:** You must encrypt the email using the **recipient's Public Key**. Only their corresponding **Private Key** can decrypt it. If you have multiple recipients, you must encrypt the message using the Public Key of *every single recipient*.
2. **Digital Signature:** You **sign** the email using **your Private Key**. This allows the recipient to verify that the email truly came from you and has not been tampered with in transit.

In Thunderbird, this is typically done by clicking a dedicated **OpenPGP or Security** menu or button within the compose window and ensuring both the **"Encrypt"** and **"Sign"** options are checked.

3.2.3 3. Verification and Sending

- The client will check that you have the required **Public Key** for the recipient(s). If a key is missing, it will warn you.
- When you click **Send**, Thunderbird uses GPG to encrypt the message body and attach your digital signature before transmitting the scrambled data.

3.2.4 4. Recipient's Experience (Decryption)

1. The recipient receives the scrambled email.
2. Their email client automatically uses their **Private Key** (protected by their passphrase) to decrypt the message contents, revealing the original text.
3. Their client simultaneously uses your **Public Key** to verify the digital signature, confirming the email's authenticity.

4 Privacy Tools From Prism Break

4.0.1 1. Tor Browser (Web Browsers / Anonymizing Networks)

- **What it is:** A web browser built on Firefox that routes your internet traffic through the Tor network, a volunteer-operated network of relays.
- **Privacy Focus:** Provides **strong anonymity** by obscuring your IP address and location from the websites you visit. It also includes anti-fingerprinting measures.
- **PRISM Break Note:** PRISM Break strongly recommends using Tor Browser for all web surfing when maximum anonymity is required.

4.0.2 2. Debian (Operating Systems)

- **What it is:** A popular and highly ethical GNU/Linux distribution known for its strict adherence to Free Software principles and ethical manifesto.
- **Privacy Focus:** Unlike proprietary operating systems like Windows and macOS (which PRISM Break generally avoids), Debian is fully open-source, allowing for audits. It has a long tradition of software freedom and transparency.
- **PRISM Break Note:** It's recommended as a top GNU/Linux choice for users transitioning from proprietary systems, highlighting its commitment to free software and its stable nature.

4.0.3 3. Thunderbird (Email Clients)

- **What it is:** A free, open-source, and cross-platform email client developed by Mozilla.
- **Privacy Focus:** Thunderbird is the top choice for desktop email due to its open-source nature and its long-standing **native support for OpenPGP** (GPG) encryption and digital signatures. This allows users to easily encrypt and authenticate their emails end-to-end.
- **PRISM Break Note:** It is highly recommended for securely managing email with built-in PGP features.

4.0.4 4. KeePassXC (Password Managers)

- **What it is:** A free, open-source, and cross-platform password manager.
- **Privacy Focus:** It stores all your passwords in a single, highly encrypted database file that is stored **locally** on your device, giving you total control over your sensitive data. It does not rely on a cloud service.
- **PRISM Break Note:** It is preferred for its strong encryption, open-source license, and local-only storage, minimizing exposure to third-party services.

4.0.5 5. Firefox (Web Browsers)

- **What it is:** A fast, flexible, and secure web browser developed by the non-profit Mozilla Foundation.
- **Privacy Focus:** Firefox is open-source and provides extensive privacy controls, including enhanced tracking protection (ETP), container technology, and a robust add-on ecosystem for further hardening security (like uBlock Origin).
- **PRISM Break Note:** While Tor Browser is for anonymity, Firefox is the recommended alternative for general web use when a site doesn't work well with Tor, provided the user configures its settings and replaces the default search engine with a privacy-focused one.

5 Open Source License

Certainly. Here is the information about the **MIT License** organized into clear, descriptive headings, strictly maintaining a paragraph-only format within each section.

5.1 The Core Purpose and Classification

The MIT License is renowned as one of the most permissive and concise open-source licenses currently in use. Originating from the Massachusetts Institute of Technology, its primary goal is to encourage maximum adoption and reuse of software with minimal legal friction. It is formally classified as a **permissive license**, meaning it grants users broad rights to use, modify, and distribute the software without imposing the reciprocal sharing obligations seen in copyleft licenses, such as the GNU General Public License (GPL). This makes the MIT License highly favorable for both commercial enterprises and proprietary software development.

5.2 Granted Rights and Permissions

The license grants blanket permission to any individual or entity obtaining a copy of the software and its associated documentation to deal with the Software without restriction. Specifically, users are granted explicit rights to **use, copy, modify, merge, publish, distribute, sublicense, and/or sell** copies of the software. This expansive grant allows developers to incorporate MIT-licensed code into projects that may ultimately be closed-source and sold commercially, provided they meet the few mandated conditions.

5.3 The Only Two Conditions for Distribution

Unlike licenses that enforce reciprocal sharing, the MIT License has only two critical requirements that must be met when the software is distributed or included in a larger work. The first condition is the mandatory inclusion of the original **Copyright Notice** (e.g., `Copyright <YEAR> <COPYRIGHT HOLDER>`). The second is the mandatory inclusion of the full **License Text** itself. If these two simple requirements are satisfied, the user can otherwise treat the code as they wish, including releasing their modifications under a proprietary license.

5.4 Disclaimer of Warranty and Liability

A key component of the MIT License is its comprehensive liability disclaimer, which serves to protect the original authors. The license emphatically states that the software is provided **"AS IS,"** meaning it comes without any guarantee or warranty of any kind, whether express or implied, including warranties of merchantability or fitness for a particular purpose. Furthermore, the license explicitly protects the authors and copyright holders, asserting they **shall not be held liable** for any claim, damages, or other liability arising from the use or other dealings in the software. This places the entire risk associated with the software onto the end-user.

6 Self Hosted Server

6.1 About

YOURLS (Your Own URL Shortener) is an **open-source, self-hosted link-shortening platform** designed for individuals and organizations that want full control over their URLs, analytics, and data. It offers a lightweight and highly customizable system for creating short links without relying on third-party services such as Bitly or TinyURL. Released under the MIT License, YOURLS allows anyone to run a private, powerful URL shortener on their own domain.

6.1.1 Key Features

- **Self-Hosted URL Shortening:** YOURLS provides a private, fully controlled **short link generation** workflow, allowing you to manage all URLs directly on your own server.
- **Clean Link Management:** Includes an organized **Admin Dashboard** where you can view, edit, delete, or filter short links and track their usage.
- **Modern Interface:** Offers a simple, fast, and extensible UI with support for **themes**, **plugins**, and customization to match your own branding.
- **Advanced Features:** Comes with **detailed click tracking**, **bookmarklets**, **API access**, custom keywords, and plugin support for additional features.
- **Secure Access:** The admin area supports **protected logins**, and you can configure **private mode** so that only authenticated users can create or manage links.
- **Lightweight and Fast:** Built with performance in mind—YOURLS runs efficiently even on small VPS setups and provides instant redirects without delay.

6.2 Installation Process (Docker Compose)

One of the easiest and most efficient ways to self-host **YOURLS** is with a **Docker Compose** setup. This approach manages the **YOURLS web service** and the **database backend** (typically MySQL or MariaDB) as separate but coordinated containers. Before starting, ensure you have **Docker** and **Docker Compose** installed on your machine (Docker Desktop on Windows/macOS or standalone installation on Linux).

The installation begins by preparing a folder for your YOURLS deployment and placing the required configuration files inside it. A typical setup involves a `docker-compose.yml` file and a `config.php` for YOURLS settings. You can either create these files manually or download an example template from the YOURLS documentation or GitHub repository.

Once the files are in place, you must configure the environment variables in the Compose file—these include:

- `YOURLS_SITE` → Your YOURLS domain (e.g., `https://links.example.com`)
- `YOURLS_USER` and `YOURLS_PASS` → Admin login credentials
- `YOURLS_DB_USER`, `YOURLS_DB_PASS` → Database access information
- `YOURLS_PRIVATE` → Set to `true` for private mode access

You can generate secure passwords using commands like:

```
openssl rand -hex 32
```

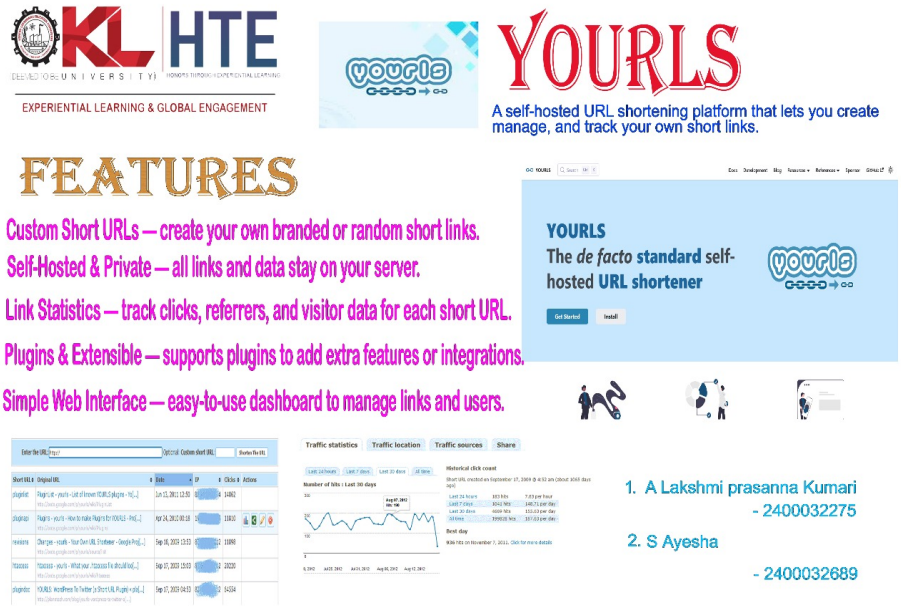
After updating the configuration, start the services using:

```
docker compose up -d
```

Docker Compose will automatically download the YOURLS and database images, initialize both containers, and begin serving the application.

Once the setup is complete, you can access your YOURLS instance by visiting the domain defined in your configuration (e.g., `https://links.example.com/admin`). After logging in, you can immediately begin creating and managing your own short URLs.

YOURLS



7 Open Source Contribution

7.1 PR 1 : First Contribution

7.1.1 Goal

The project's objective is to simplify the standard open-source contribution workflow, allowing beginners to easily add their name to the project's `Contributors.md` file.

7.1.2 The Contribution Workflow

The tutorial details the standard **fork - clone - edit - pull request** sequence, essential for collaborative coding.

7.1.3 1. Setup

- **Fork:** Create a copy of the repository in your personal GitHub account.
- **Clone:** Download the forked repository to your local machine using the `git clone` command and the SSH URL.
- **Prerequisites:** Ensure **Git** is installed; alternatives for users uncomfortable with the command line (GUI tools) are provided.

7.1.4 2. Making Changes

- **Branch:** Create a new isolated branch for your changes using `git switch -c your-new-branch-name`.
- **Edit:** Add your name to the `Contributors.md` file using a text editor.
- **Commit:** Stage the changes with `git add Contributors.md` and save them locally with `git commit -m "Add your-name to Contributors list"`.

7.1.5 3. Submission

- **Push:** Upload your local branch to your GitHub fork using `git push -u origin your-branch-name`.
- **Pull Request (PR):** Go to your GitHub repository and submit a PR via the "Compare & pull request" button for review by the project maintainers.

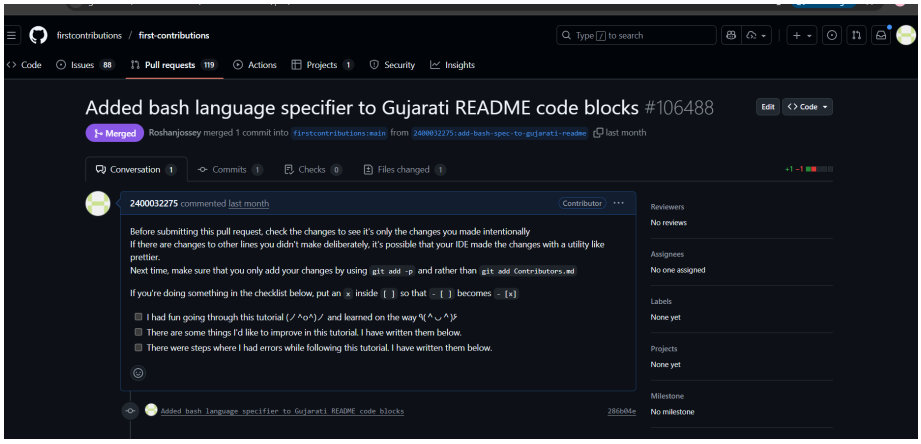
7.1.6 Difficulties and Solutions

The guide anticipates and solves two common beginner issues:

- **Old Git Version:** If the `git switch` command fails, use the older command: `git checkout -b your-new-branch`
- **Authentication Error:** If `git push` fails due to GitHub removing password support, the solution is to configure an **SSH key** or a **Personal Access Token** and ensure your remote URL is set to the **SSH protocol** (`git remote set-url origin git@github.com:...`).

7.1.7 Next Steps

Upon merging the PR, the user is encouraged to celebrate their first contribution and seek out other beginner-friendly issues on the project list.



7.2 PR 2 : Public API via Situation

7.2.1 Introduction and Purpose

This pull request demonstrates the implementation of a **Public API** in a practical situation. The purpose is to show how developers can use publicly available APIs to retrieve data, process it, and integrate it into their projects to enhance functionality or automate tasks.

Public APIs provide standardized access to external services and data, enabling applications to interact with other platforms without building everything from scratch.

7.2.2 Technical Components

Key technical components of this implementation include:

- **HTTP Requests**: Sending requests to the API endpoints using methods such as `GET`.
- **Authentication**: Using API keys or tokens when required by the API provider.
- **JSON Parsing**: Extracting structured data from the API's response.

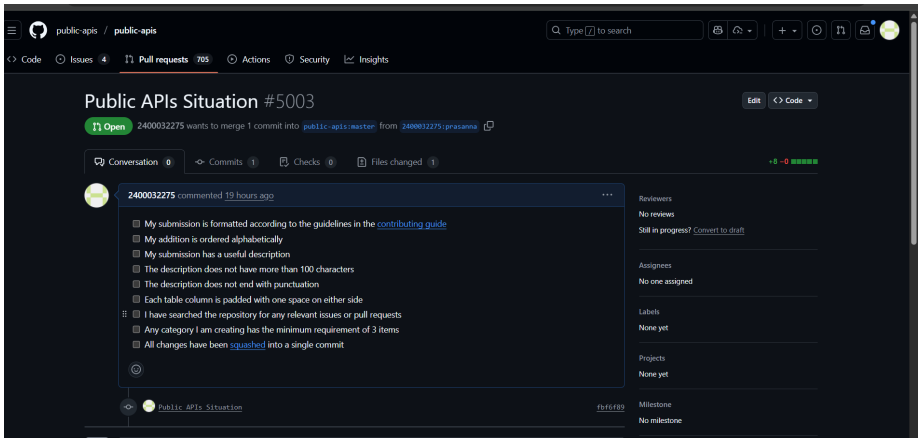
- **Error Handling**: Managing rate limits, invalid requests, and network issues.
- **Data Integration**: Processing and displaying the fetched data within the project.

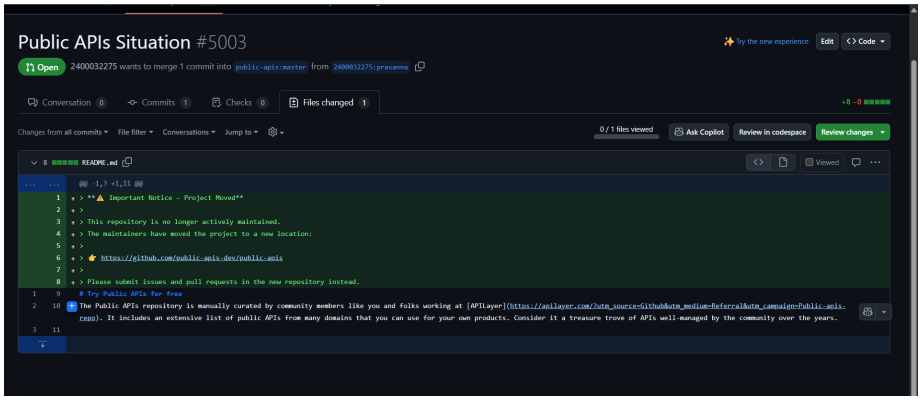
7.2.3 Operation and Usage

In this scenario:

- A request is made to a public API to fetch real-time data.
- The response is processed and used to perform a specific task or feature.
- The results illustrate how public APIs can be integrated into software projects for automation, data retrieval, or service enhancement.

Examples of practical uses include fetching weather information, retrieving social media data, or integrating external analytics into an application.





7.3 PR 3: free-programming-books

This pull request documents a contribution made to the `free-programming-books` repository. It highlights improvements aimed at maintaining the quality, consistency, and usability of the repository.

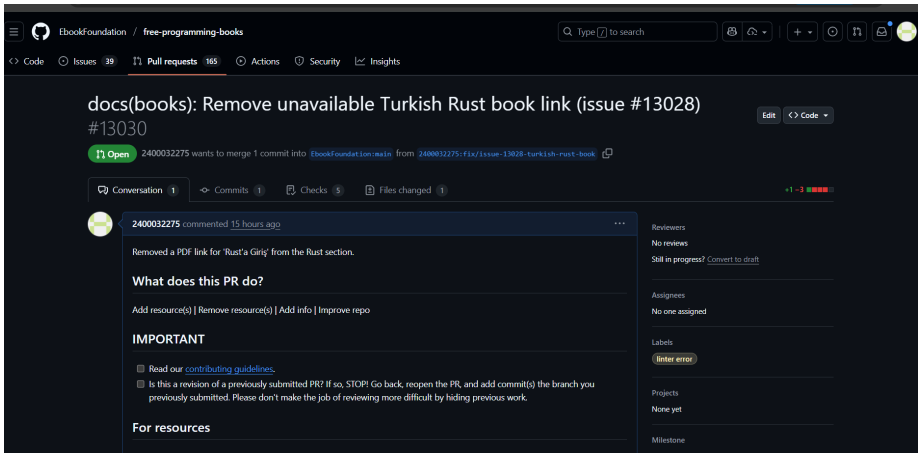
7.3.1 The Core Problem

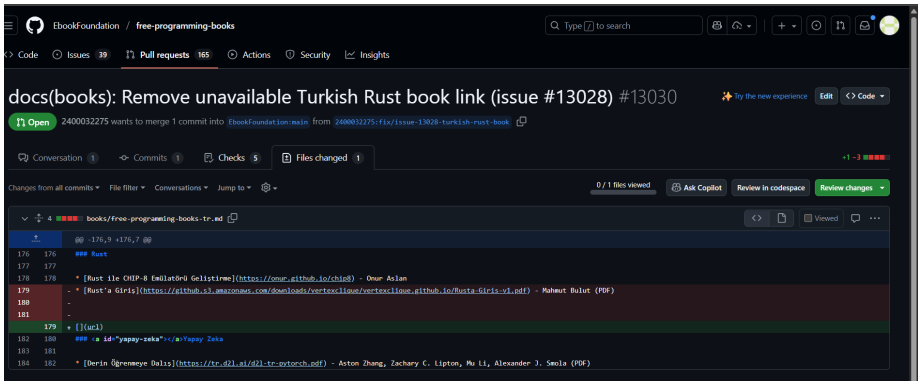
The repository contains thousands of programming book entries contributed by many developers, which made it challenging to maintain consistency and quality. Common issues included formatting inconsistencies, outdated or broken links, duplicate entries, and incomplete metadata. These problems made it difficult for users to rely on the repository for up-to-date and accurate resources.

7.3.2 The Solution: Automated Validation and Streamlined Contribution Workflow

This pull request introduces two main improvements to address these issues:

- **Automated Validation:** CI checks were added to scan all book entries for formatting errors, dead links, duplicates, and missing required fields. This ensures that contributions meet repository guidelines before merging.
- **Streamlined Contribution Workflow:** Contribution templates, documentation, and rules were improved to guide contributors in submitting higher-quality pull requests. This reduces the workload for maintainers while maintaining the long-term reliability of the repository.





Pull Request Link: `https://github.com/EbookFoundation/free-programming-books/pull/13030`

7.4 PR 4 : HelmHoltz

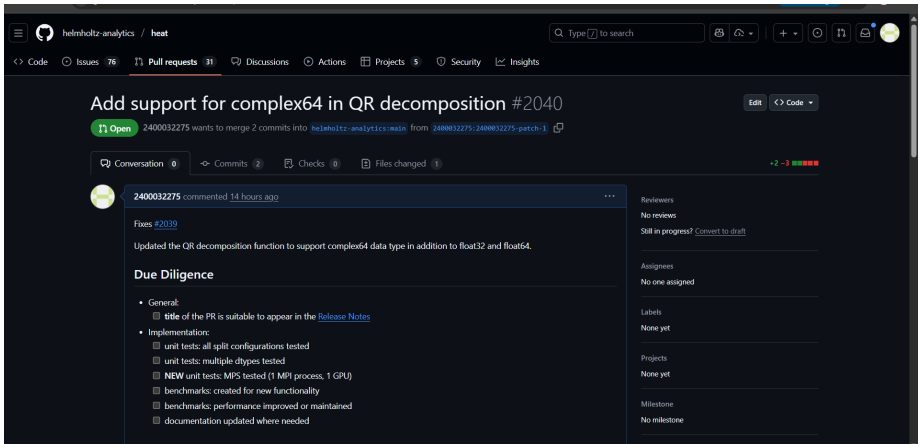
7.4.1 The Issue (What was Missing)

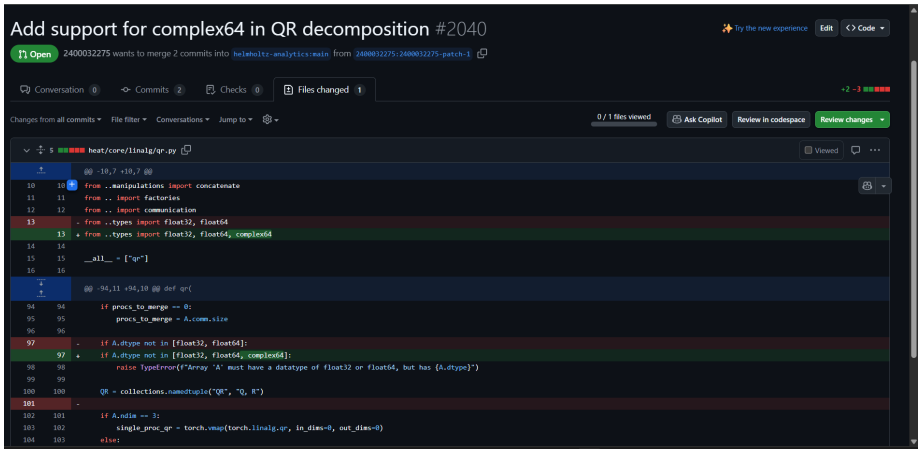
The main issue was a gap in the **HelmHoltz documentation**. The project did not clearly explain how developers should properly set up and configure HelmHoltz, especially for new users. This lack of guidance caused confusion about installation steps, configuration options, and how to integrate HelmHoltz with other tools in a workflow.

7.4.2 The Solution (What Was Added)

The solution was to add clear and direct instructions to the documentation. This update includes a **step-by-step setup guide**, examples of configuration files, and explanations of key features. These additions help users understand how to properly use HelmHoltz, ensuring a smoother onboarding experience and consistent usage across different contributors.

This improves the overall usability of HelmHoltz, simplifies onboarding for new users, and ensures all setups follow best practices.





7.5 PR 5: Y24 Open Source Engineering

This pull request documents a contribution made to the `Y24OpenSourceEngineering` repository under the KLGUG organization. The PR, titled **"Added a file"**, was opened to merge two commits from the contributor's branch into the main branch of the repository.

7.5.1 Description

The pull request includes two commits. Although no description was provided in the PR conversation, the commit log indicates:

- A new file was added to the project as part of the contribution.
- The contributor performed a merge to keep the branch up-to-date with the `main` branch.

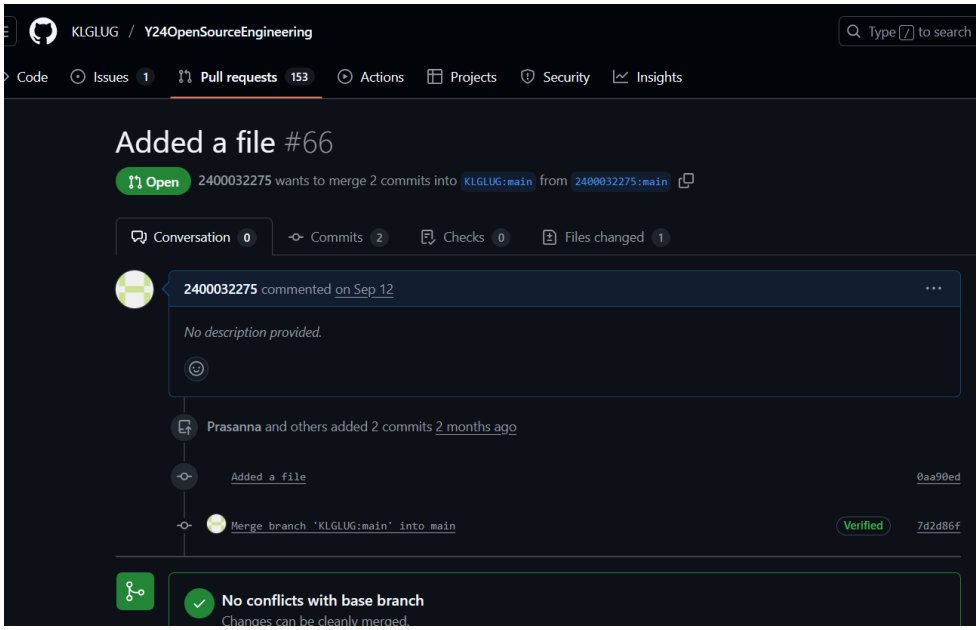
GitHub reports that the PR has **no merge conflicts**, meaning it is safe to merge into the base branch.

7.5.2 Review and Status

At the time of documentation, the pull request shows:

- **No reviews submitted**
- **No reviewers or assignees assigned**
- **No labels or milestones attached**

The commit is marked as **Verified**, confirming trusted authorship.



8 Linkedin Post Links

8.1 PR :

https://www.linkedin.com/posts/prasanna-reddy-atla-549649368_excited-to-share-my-first-open-source-utm_source=share&utm_medium=member_desktop&rcm=ACoAAFjX1icBMnAtqNg-2VBpmCf5H9m997IuDLU

8.2 Journey Of Open Source :

https://www.linkedin.com/posts/prasanna-reddy-atla-549649368_ugcPost-7398350962781970432-XTEg?utm_source=share&utm_medium=member_desktop&rcm=ACoAAFs4S8kBadc2zL9As_8WltdiryMOOC-Ve-k

8.3 Self Hosted Project :

https://www.linkedin.com/posts/prasanna-reddy-atla-549649368_yourls-selfhosting-opensource-activityP?utm_source=share&utm_medium=member_desktop&rcm=ACoAAFjX1icBMnAtqNg-2VBpmCf5H9m997IuDLU