OPEN SOURCE ENGINEERING

Student ID: 2400040026

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

Jitsi Meet is an open-source, self-hosted video conferencing platform that makes online meetings easy and secure. It provides a clean and user-friendly interface for real-time audio and video communication, without depending on any paid or closed-source services. Built to give users full freedom and control, it is released under the Apache 2.0 License, which allows anyone to use, modify, and host it on their own servers.

6.1.1 Key Features

- **Video Conferencing Approach:** Jitsi Meet follows a simple and secure **video meeting** method, allowing users to join rooms instantly without needing accounts.
- **Organized Meeting Structure:** A self-hosted setup can include different **domains - rooms - participants - features** such as chat, screen share, and recording.
- **User-Friendly Interface:** It provides a clean **Dark Mode**, a **Minimal Wide Layout** for clear video tiles, and **Collapsable Side Panels** to give more space during meetings.
- **Power Features:** Jitsi includes tools like **Moderator Controls**, **Shortcut Keys**, and **Multitasking Features** that allow users to chat, manage participants, and share their screen at the same time.
- **Secure Access:** Supports **JWT / SSO integration** for protected meeting access and controlled user authentication in self-hosted environments.
- **Realtime Communication:** Built for instant audio/video updates, so all actions (mute, chat, join/leave) appear immediately without reloading the meeting page.

6.2 Installation Process (Docker Compose)

The recommended and most convenient way to self-host **Jitsi Meet** is by using **Docker Compose**, which manages all required services such as the **web interface**, **Jicofo**, **JVB (Video Bridge)**, and **Prosody** in a single, organized multi-container setup. Before starting the installation, make sure that **Docker** is installed on your system and that **Docker Compose** is available (included in Docker Desktop or installed separately on Linux).

The setup begins by **downloading the official configuration files**. You can use the `curl` command to fetch the required `.env` and `docker-compose.yml` files from the Jitsi GitHub repository and save them inside your deployment folder. After downloading, you must perform an important configuration step: **editing the environment variables** inside the `.env` file. You need to set the `HTTP_PORT` or `HTTPS_PORT` depending on your deployment, update the `PUBLIC_URL` to match your domain or public IP (e.g., `https://meet.example.com`), and generate secure random passwords for all Jitsi core services such as `JICOFO_COMPONENT_SECRET`, `JICOFO_AUTH_PASSWORD`, and `JVB_AUTH_PASSWORD`. These passwords can be generated using commands like `openssl rand -hex 32`.

Once the environment settings are correctly configured, the next step is to **download the images and start the containers**. Running the command `docker compose up -d` will automatically pull all required Jitsi Meet images and launch every service in detached mode (`-d`). Docker Compose ensures that the web interface, signaling server, and video bridge all work together smoothly.

After the services are running, **accessing Jitsi Meet** is simple: open your browser and visit the URL you configured in the `PUBLIC_URL` field. For local setups, the default access point is `http://localhost:8000`. Once loaded, you can immediately start creating or joining meeting rooms, as Jitsi Meet does not require a default username or password for basic usage unless you enable authentication in the `.env` configuration.

Jitsi Meet Resources

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.

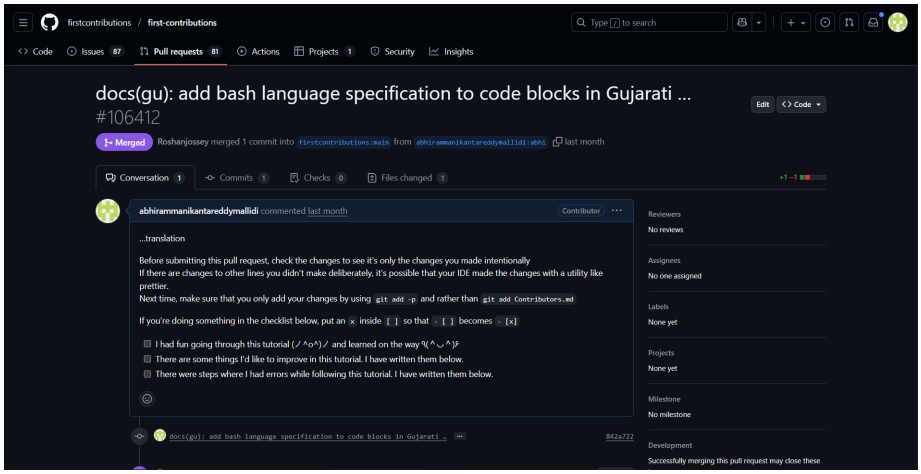
7.1.6 Difficulties and Solutions

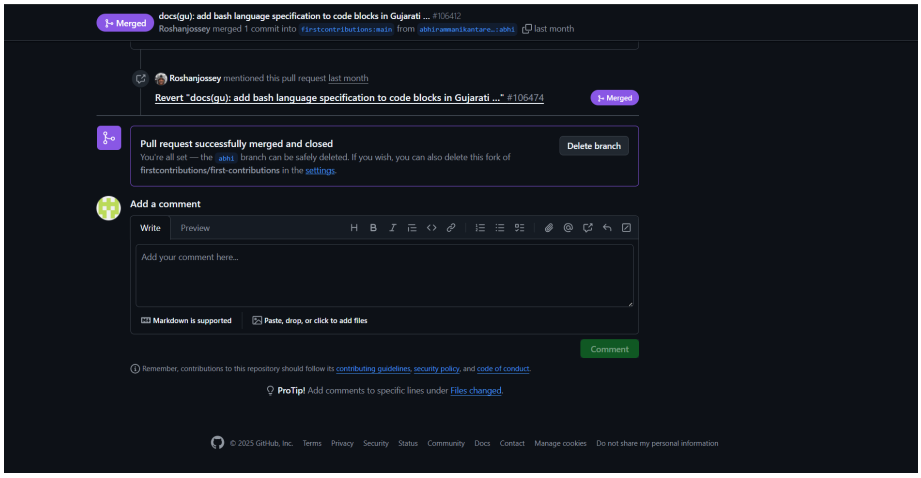
The guide anticipates and solves two common beginner issues:

- **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 : freeCodeCamp

freeCodeCamp is a free, open-source learning platform designed to help people learn programming through hands-on practice, projects, and certifications. It works like a complete "learning operating system" on the internet, offering thousands of coding lessons, interactive challenges, and full-length projects that learners can access from anywhere. It serves multiple important roles: mainly as a self-paced learning environment, a community-driven platform for contributing to open-source projects, and a career-building resource for new developers. It acts as a strong, transparent, and community-first alternative to paid learning platforms.

7.2.1 Licensing and Self-Hosting Options

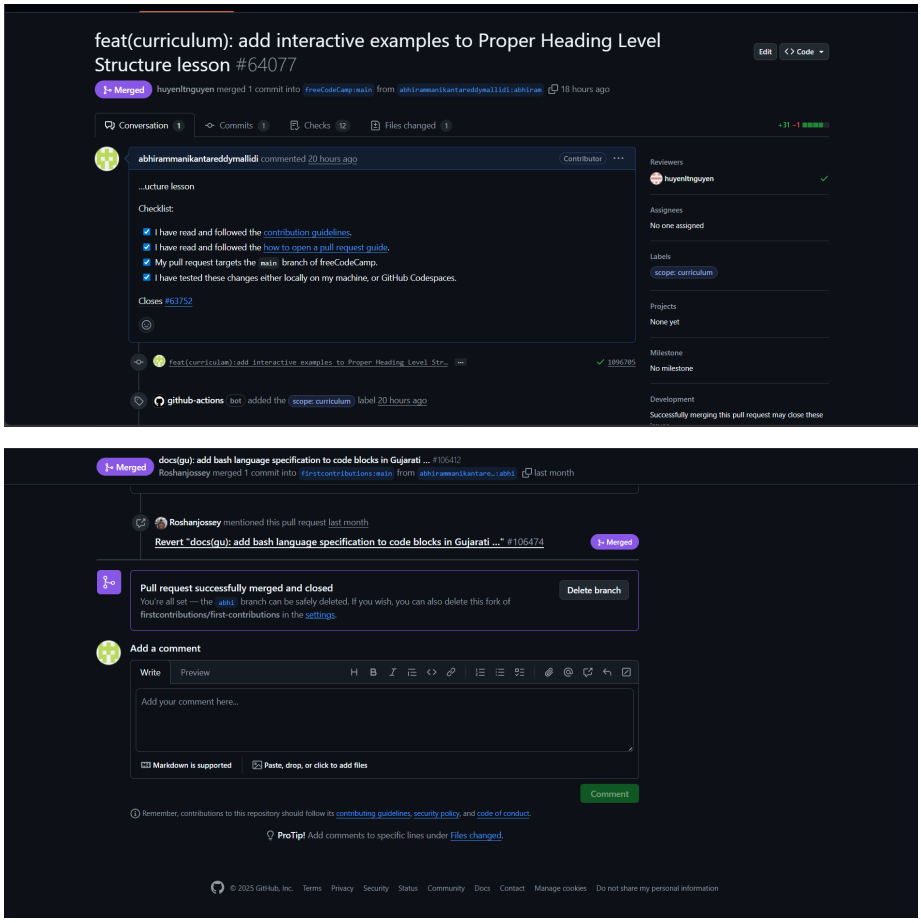
The entire freeCodeCamp project is released under the **BSD-3-Clause License**. This license is very permissive, allowing individuals and organizations to use, change, and redistribute the code with minimal restrictions. The platform strongly supports community contributions and provides clear documentation for setting up a local development environment.

For deployment and development, freeCodeCamp offers detailed instructions for **Local Development** using Node.js (version 18+ is recommended). Developers can run the platform locally for testing, curriculum improvements, or contributing to the codebase. Self-hosting is possible using **Docker** and **Docker Compose**, making it easy to run the platform on Linux, macOS, and Windows systems. Requirements include Node.js, MongoDB, and basic system resources for smooth operation. For direct access without hosting, the live learning platform is available at freeCodeCamp.org.

7.2.2 Community and Support

freeCodeCamp maintains a strong and active global community with many ways to connect and participate. Users and contributors can interact through the official Forum, GitHub, Discord, and social media platforms such as X (Twitter). For issues related to the platform, users are encouraged to open bugs, feature requests, or pull requests through the project's GitHub repository. Curriculum-related discussions also happen within GitHub issues and pull requests.

For support, documentation, and contribution guidelines, freeCodeCamp provides extensive resources. Security-related concerns can be privately shared using the contact methods provided in the repository's documentation, ensuring responsible and timely handling of sensitive matters.



7.3 PR 3 : Y24 Open Source Engineering

7.3.1 Introduction and Purpose

Jitsi Meet is a modern, **open-source video conferencing platform** designed to provide free, secure, and high-quality online meetings. Its primary objective is to offer users complete privacy by running their own instance independently of third-party services, making it a powerful solution for **self-hosting** video calls, online classes, and collaborative communication with full control over data and infrastructure.

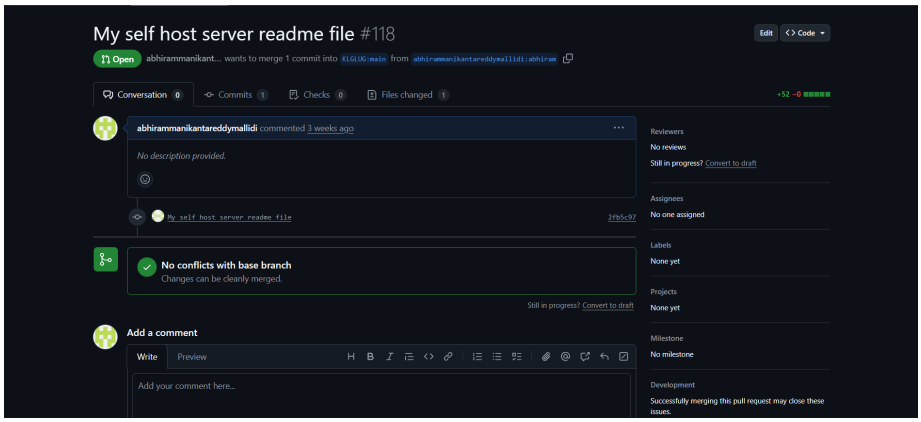
7.3.2 Technical Components

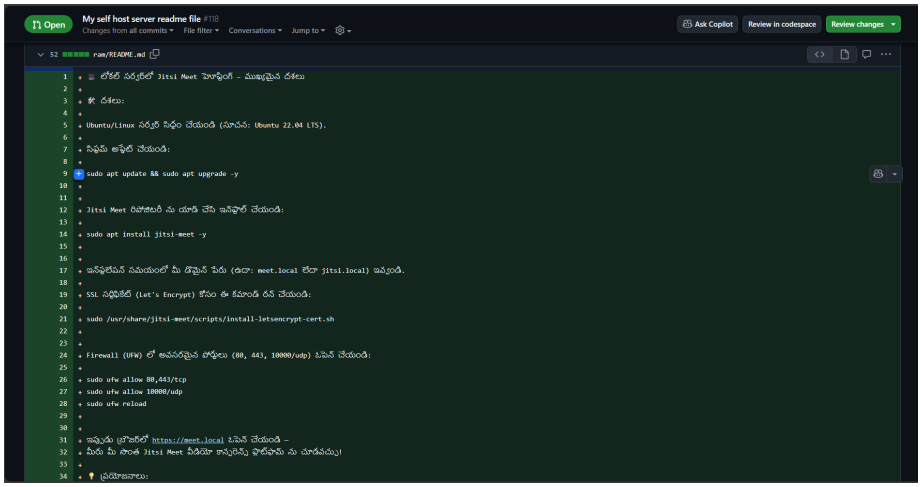
A self-hosted Jitsi Meet setup relies on a **Linux-based server environment**, typically running distributions such as **Ubuntu Server** or **Debian**. The system includes essential components such

as the **Jitsi Videobridge**, **Prosody** (for XMPP/JWT authentication), and **Jicofo**, all of which work together to manage video streams, authentication, and conference coordination. Network requirements include stable **connectivity**, proper firewall configuration, and support for **TLS/SSL certificates** (commonly via Let's Encrypt).

7.3.3 Operation and Usage

Jitsi Meet operates by running its service stack on the Linux server and configuring the corresponding domains, ports, and SSL certificates. Access is provided to users through a browser or mobile app using the server's domain name. The platform is ideal for secure **video conferences**, hosting **online classes**, conducting **team meetings**, or enabling private communication environments. This gives users full administrative control, improved privacy, and a hands-on experience in managing real-time communication systems.





7.4 PR 4: public-apis

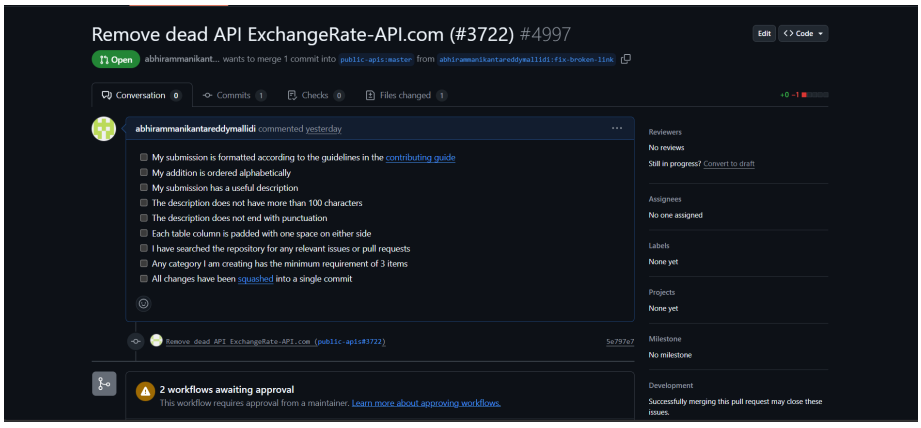
Here is a short description of the Pull Request, organized into paragraphs with headings.

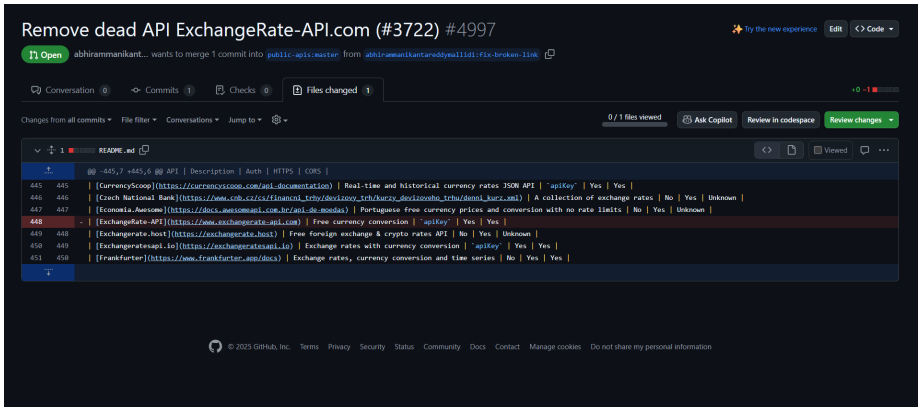
7.4.1 The Core Problem

The main issue addressed in this Pull Request was the repository's growing difficulty in maintaining consistency and quality across thousands of API entries. Because contributions come from many developers, the project suffered from formatting inconsistencies, outdated links, broken APIs, and duplicate submissions. Additionally, the manual review process placed a heavy burden on maintainers, making it time-consuming to validate each contribution and ensure it met the project's strict quality standards.

7.4.2 The Solution: Automated Validation and Streamlined Contribution Workflow

This Pull Request introduces two major improvements to solve these challenges. First, it implements Automated Validation by adding enhanced CI checks that scan API entries for formatting errors, dead links, duplicates, and missing required fields. This ensures that contributions remain clean, valid, and compliant with project guidelines before reaching maintainers. Second, it streamlines the overall Contribution Workflow by updating templates, improving documentation, and adding clearer contribution rules. These changes guide contributors to submit higher-quality pull requests, reducing maintainer workload and improving the long-term reliability of the public-apis repository.





7.5 PR 5 : Emoji-Log

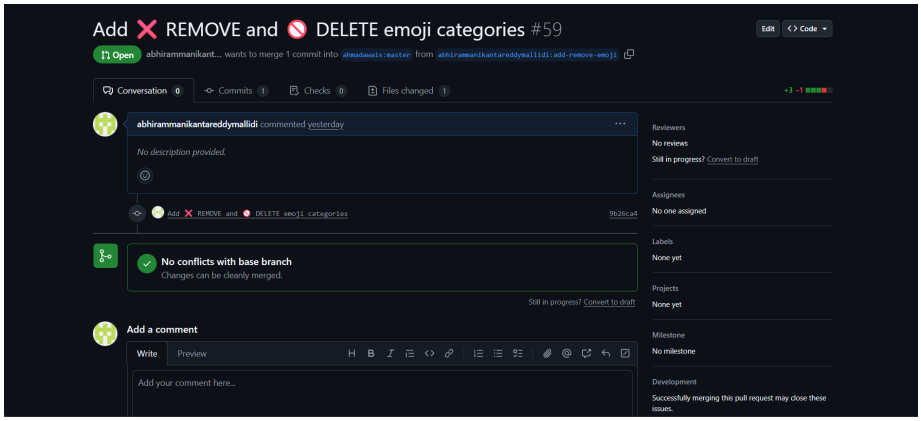
7.5.1 The Issue (What was Missing)

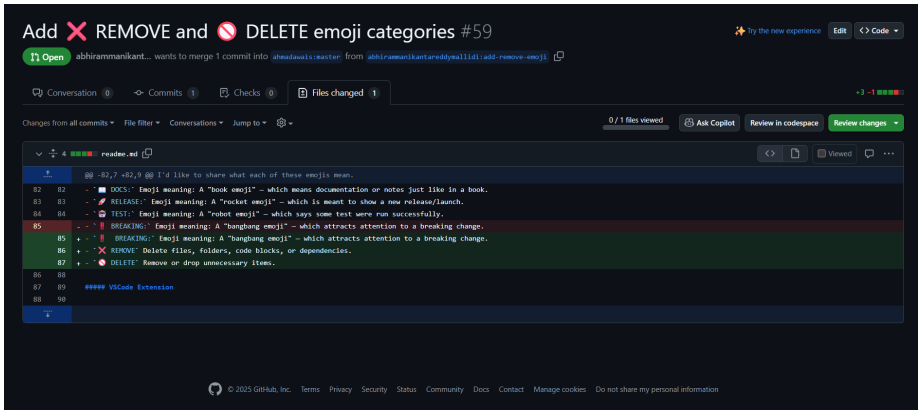
The main issue was a gap in the **Emoji-Log documentation**. The project did not clearly explain how developers should properly format commit messages when using Emoji-Log, especially for people who were new to the style. This lack of guidance caused confusion about which emojis to use, how to structure the commit message, and how to keep the log consistent across different contributors.

7.5.2 The Solution (What Was Added)

The solution was to add clear and direct instructions to the documentation. This update includes a **detailed commit style guide**, examples of correctly formatted Emoji-Log commit messages, and explanations of each emoji category. These additions help users understand when and how to use specific emojis, ensuring consistent commit history and easier collaboration across the project.

This improves the overall usability of Emoji-Log, makes onboarding simpler for new contributors, and ensures that all commit messages follow the same clean and readable format.





8 Linkedin Post Links

8.1 PR :

https://www.linkedin.com/posts/mallidi-abhiram-manikanta-reddy-087219357_excited-to-share-my-first-
utm_source=share&utm_medium=member_desktop&rcm=ACoAAFjX1icBMnAtqNg-2VBpmCf5H9m997IuDLU

8.2 Journey Of Open Source :

https://www.linkedin.com/posts/mallidi-abhiram-manikanta-reddy-087219357_activity-73982502673951170
utm_source=share&utm_medium=member_desktop&rcm=ACoAAFjX1icBMnAtqNg-2VBpmCf5H9m997IuDLU

8.3 Self Hosted Project :

https://www.linkedin.com/posts/mallidi-abhiram-manikanta-reddy-087219357_devops-selfhosting-jitsime
utm_source=share&utm_medium=member_desktop&rcm=ACoAAFjX1icBMnAtqNg-2VBpmCf5H9m997IuDLU