OPEN SOURCE ENGINEERING

Student Name: S. Surya Ganesh Reddy
Student ID: 2400030085
Course Name: Open Source Engineering
Academic Year: 2025-2026

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 Used

I used the **GNU GENERAL PUBLIC LICENSE** license for my project.

Why This License?

- Allows anyone to use, modify, and distribute.
- Encourages open collaboration.
- Compatible with open-source communities.

License Text

Open Source License Certainly. Here is the information about the **GNU General Public License (GPL)** organized into clear, descriptive headings, strictly maintaining a paragraph-only format within each section.

5.0.1 The Core Purpose and Classification

The GNU General Public License (GPL), developed by the Free Software Foundation (FSF) under the leadership of Richard Stallman, is one of the most influential and widely used open-source licenses in the world. The core purpose of the GPL is to guarantee end-users the essential software freedoms: the freedom to run, study, share, and modify the software. It is formally classified as a **strong copyleft license**, meaning any distributed derivative work must also be licensed under the GPL. This copyleft mechanism ensures that improvements and modifications remain free for all users, preserving the long-term openness and accessibility of the software ecosystem.

5.0.2 Granted Rights and Permissions

The GPL grants users comprehensive rights to copy, modify, distribute, and run the software for any purpose, without restriction. Users can access and change the source code, integrate it into their own projects, and even distribute modified versions, provided they comply with the copyleft rules. These permissions promote collaborative improvement and foster a global community dedicated to maintaining free and open software. Unlike permissive licenses, however, the GPL ensures that any redistributed work that incorporates GPL code must also remain open-source, maintaining an unbroken chain of software freedom.

5.0.3 The Conditions and Copyleft Requirements

The GPL introduces more rigorous conditions than permissive licenses, designed to uphold its philosophy of software freedom. The most important condition is the requirement that any derivative or redistributed version of GPL-licensed software must also be licensed under the **same GPL license**, including the obligation to release the complete corresponding source code. When distributing binaries or modified versions, distributors must make the source code available in an accessible form. Additionally, the license text and original copyright notice must be included with every distribution. These copyleft conditions ensure that all improvements remain free and that no one can take GPL-licensed software and turn it into proprietary software.

5.0.4 Disclaimer of Warranty and Liability

Like most open-source licenses, the GPL includes a strong disclaimer of warranty to protect authors from legal liability. The software is provided **"AS IS"**, without any express or implied warranties, including merchantability, fitness for a particular purpose, or non-infringement. The license clarifies that the authors and copyright holders cannot be held responsible for damages, losses, or legal claims arising from the use, modification, or distribution of the software. This places the responsibility of use entirely on the recipient, ensuring that developers can contribute freely without fear of legal repercussions.

6 Self-Hosted Server

I self-hosted the service: **My Tiny To-do**.

About the Service

MyTinyTodo is a simple, open-source, self-hosted to-do list and task management application. It allows users to create, organize, and track tasks from a clean web-based interface. Because it is lightweight and built using PHP and MySQL/sqlite, it can run on almost any server with minimal setup.

It supports features like multiple task lists, tags, due dates, priority levels, and AJAX-based quick editing. Users can also enable public sharing of lists, integrate with mobile browsers, and customize themes. Since it is self-hosted, users have full control over their data and can modify the source code as needed. MyTinyTodo is often used for personal productivity, small teams, and simple project/task management due to its simplicity and flexibility.

Installation Commands

```
sudo apt update && sudo apt upgrade -y
sudo apt install apache2 -y
sudo systemctl enable apache2 && sudo systemctl start apache2
sudo apt install php php-mbstring php-mysql php-json php-curl php-gd php-xml unzip -y
sudo systemctl restart apache2
sudo apt install mysql-server -y
sudo systemctl enable mysql && sudo systemctl start mysql
sudo mysql -e "CREATE DATABASE mytinytodo;"
sudo mysql -e "CREATE USER 'mttuser'@'localhost' IDENTIFIED BY 'StrongPassword123!';"
sudo mysql -e "GRANT ALL PRIVILEGES ON mytinytodo.* TO 'mttuser'@'localhost';"
sudo mysql -e "FLUSH PRIVILEGES;"
cd /var/www/html
sudo wget https://www.mytinytodo.net/download/mytinytodo-latest.zip -O mytinytodo.zip
sudo unzip mytinytodo.zip
sudo rm mytinytodo.zip
sudo mv mytinytodo*/ mytinytodo
sudo chown -R www-data:www-data /var/www/html/mytinytodo
sudo chmod -R 755 /var/www/html/mytinytodo
cd /var/www/html/mytinytodo
sudo cp config.php.example config.php
sudo sed -i "s/'user' => ''/'user' => 'mttuser'/g" config.php
sudo sed -i "s/'pass' => ''/'pass' => 'StrongPassword123!'/g" config.php
sudo sed -i "s/'name' => ''/'name' => 'mytinytodo'/g" config.php
sudo sed -i "s/'type' => 'sqlite'/'type' => 'mysql'/g" config.php
sudo a2enmod rewrite
sudo systemctl restart apache2
```

Screenshot

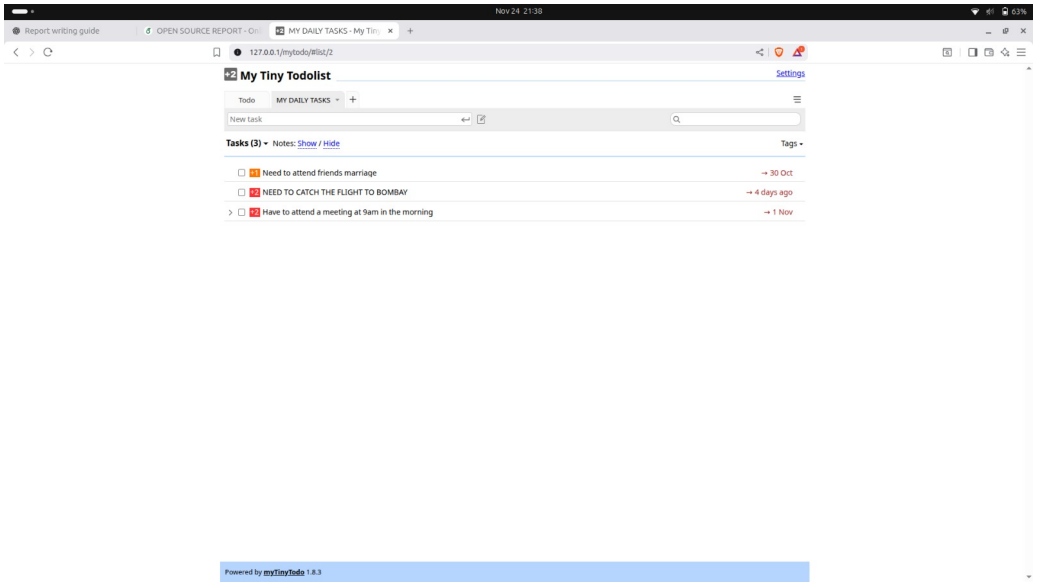


Figure 1: Self Hosted Server Running

Localized Document (Translated)

Self Hosting Localized document: Self Hosting

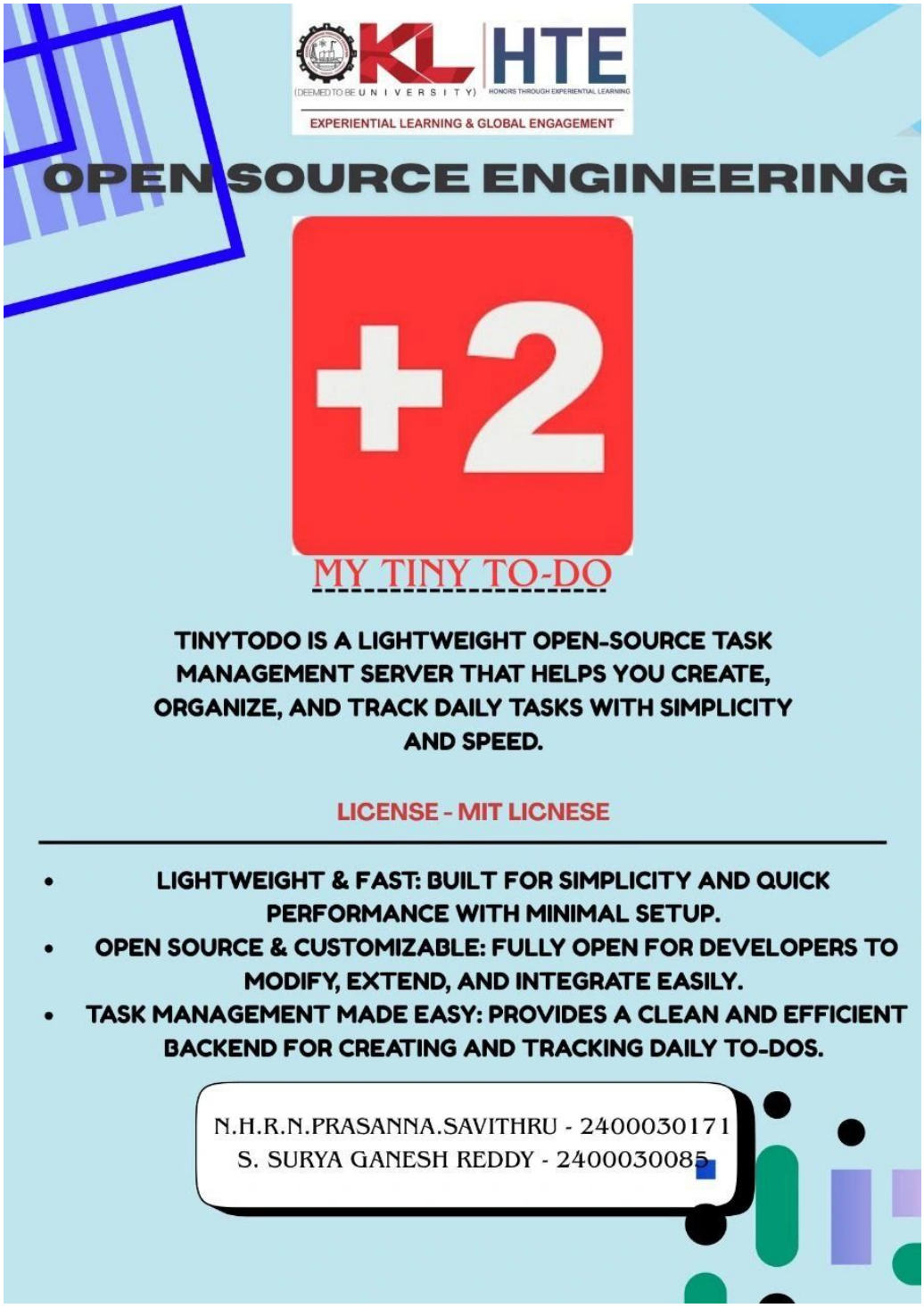


Figure 2: Poster

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.

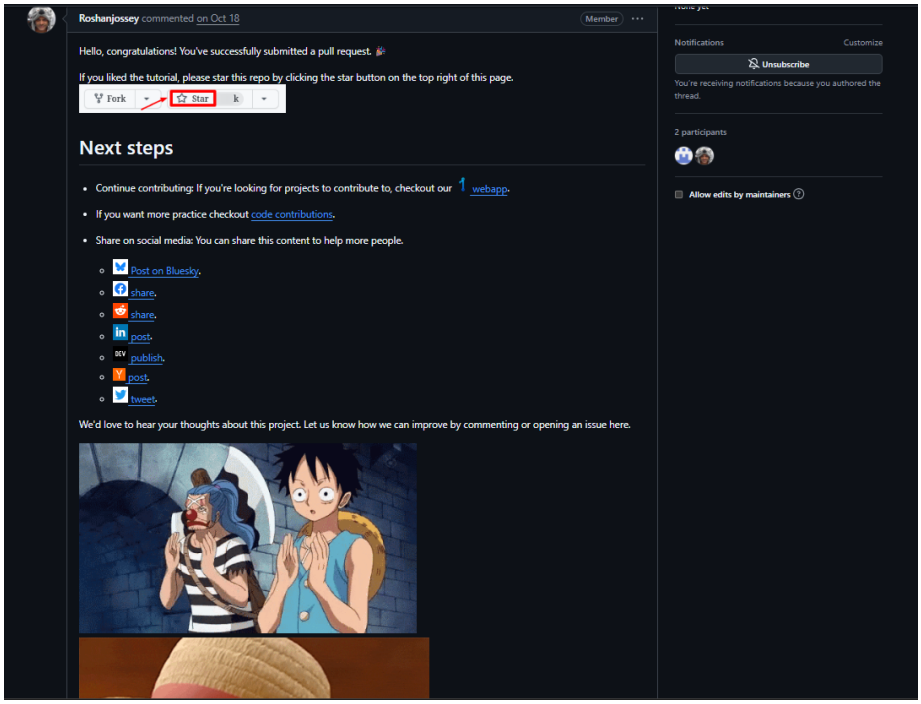
7.1.6 Difficulties and Solutions

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 : Puter

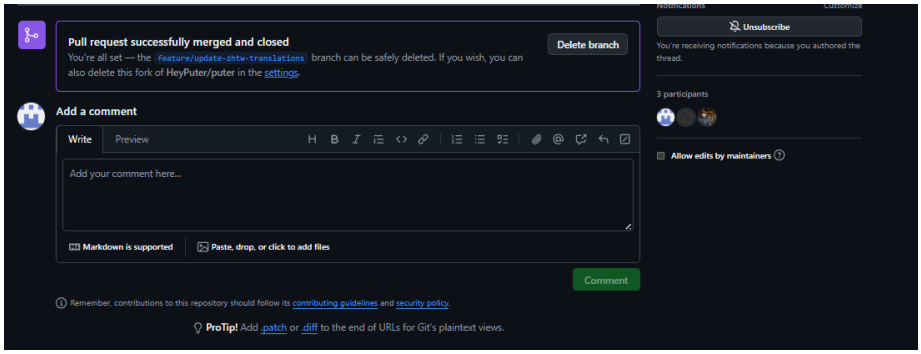
Puter is an advanced, free, and open-source platform, effectively functioning as an "Internet OS" that is both exceptionally fast and highly extensible. It is designed to serve multiple critical roles: primarily as a privacy-first **personal cloud** for securely keeping all user files, apps, and games accessible from anywhere; as a robust platform for building and publishing websites and web applications; and as a versatile **remote desktop environment** for managing servers and workstations. It is positioned as a powerful, feature-rich alternative to services like Dropbox and Google Drive, built on principles of open-source transparency.

7.2.1 Licensing and Self-Hosting Options

The entire Puter project is distributed under the **AGPL-3.0 license**. This license is highly protective, requiring that any individual or entity who runs the modified software over a network must make the corresponding source code publicly available. For deployment, the project strongly emphasizes self-hosting, providing detailed instructions for several methods: **Local Development** using Node.js (version 20.19.5+ is required) for immediate testing, and easy, reliable deployment using **Docker** or **Docker Compose** on Linux, macOS, and Windows systems. Users require a minimum of 2GB of RAM to run the system smoothly. For immediate access without self-hosting, a live hosted version is also available at Puter.com.

7.2.2 Community and Support

Puter maintains a strong connection with its community through various channels for support and contribution. Users can engage directly with maintainers via Discord, Reddit, and X (Twitter). For technical issues, the community is encouraged to report bugs or request features by opening an issue on the project's repository. Specific security concerns can be privately addressed by emailing `security@puter.com`, ensuring a responsive and accountable support structure.



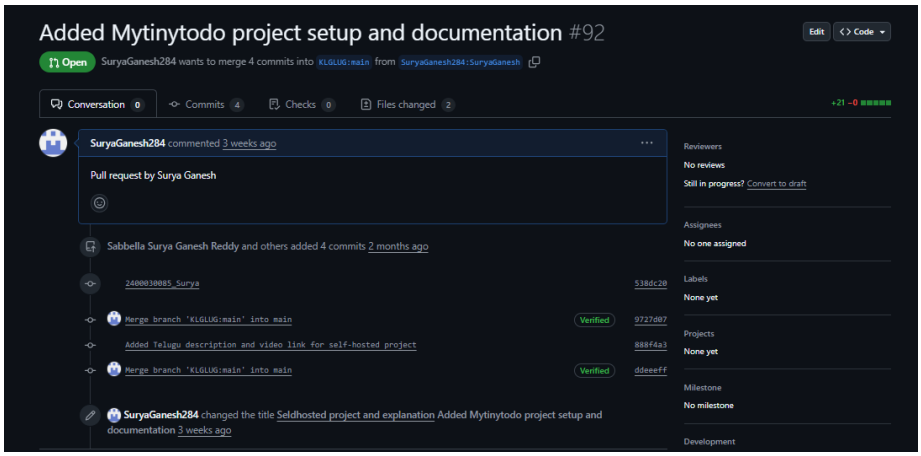
7.3 PR 3 : Y24 Open Source Engineering

7.3.1 Introduction and Purpose

4GA Boards are modern **Embedded System Boards** designed to function as cost-effective **Self-Hosting Servers**. Their primary objective is to empower users to maintain their data independently of external cloud services, thereby maximizing **data privacy and security**. This provides an opportunity for individuals to host small-scale web or IoT (Internet of Things) applications with full control over the infrastructure.

7.3.2 Technical Components

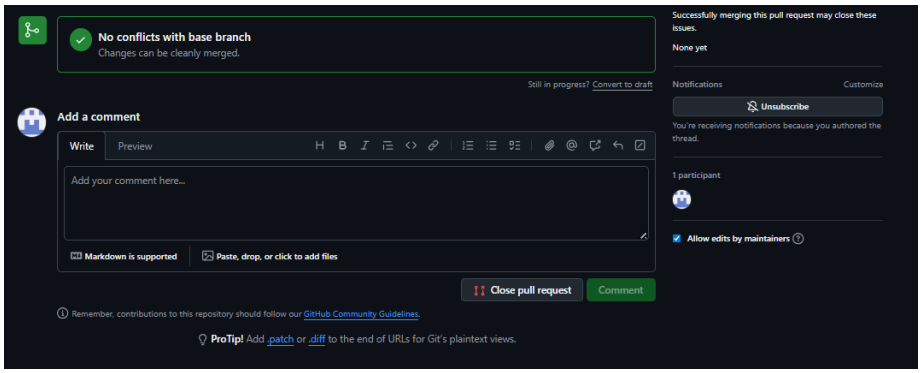
The core server setup relies on the 4GA Board, which features an **ARM Cortex-based processor**. Essential components include Micro SD/SSD storage, network **connectivity** via Ethernet or Wi-Fi, and a dedicated **power supply**. The entire system is built upon a Linux-based operating system, such as **Ubuntu Server** or **Debian**, which provides the necessary server environment.



7.3.3 Operation and Usage

The board operates by running a Linux server OS and configuring standard web server software (like Apache or Nginx). Access is typically granted locally via the network or publicly via a domain

name. These systems are ideal for **IoT Data Collection**, managing small **database systems**, hosting personal websites, or creating a private file-sharing and **cloud backup server**, offering users complete management control and valuable practical experience.



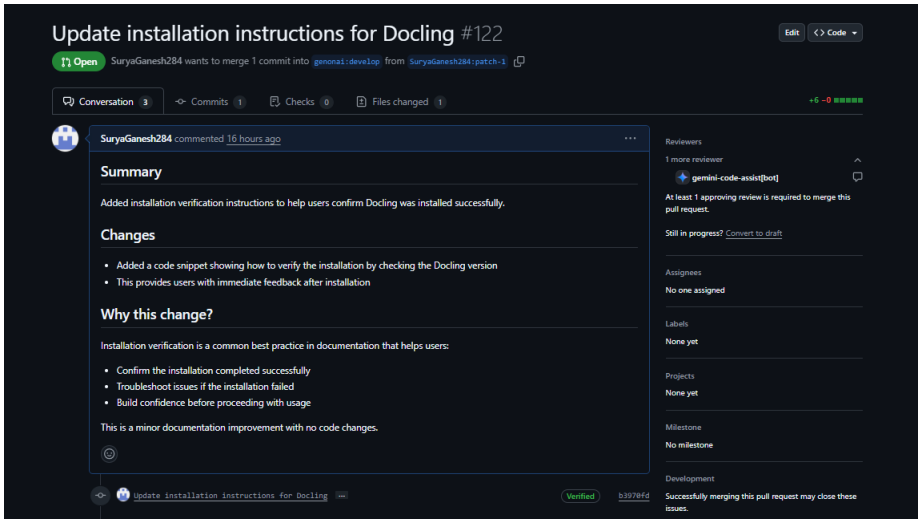
7.4 PR 4 : GenonAI / doc_parser

7.4.1 Introduction and Purpose

The **GenonAI doc_parser** project is an intelligent **document processing system** designed to automatically extract, analyze, and structure information from unorganized text files. Its primary purpose is to convert raw documents into clean, machine-readable formats, enabling efficient downstream tasks such as **semantic search**, **automated summarization**, and **AI-driven data retrieval**. This provides developers and organizations with a streamlined, privacy-focused alternative to external document-processing platforms.

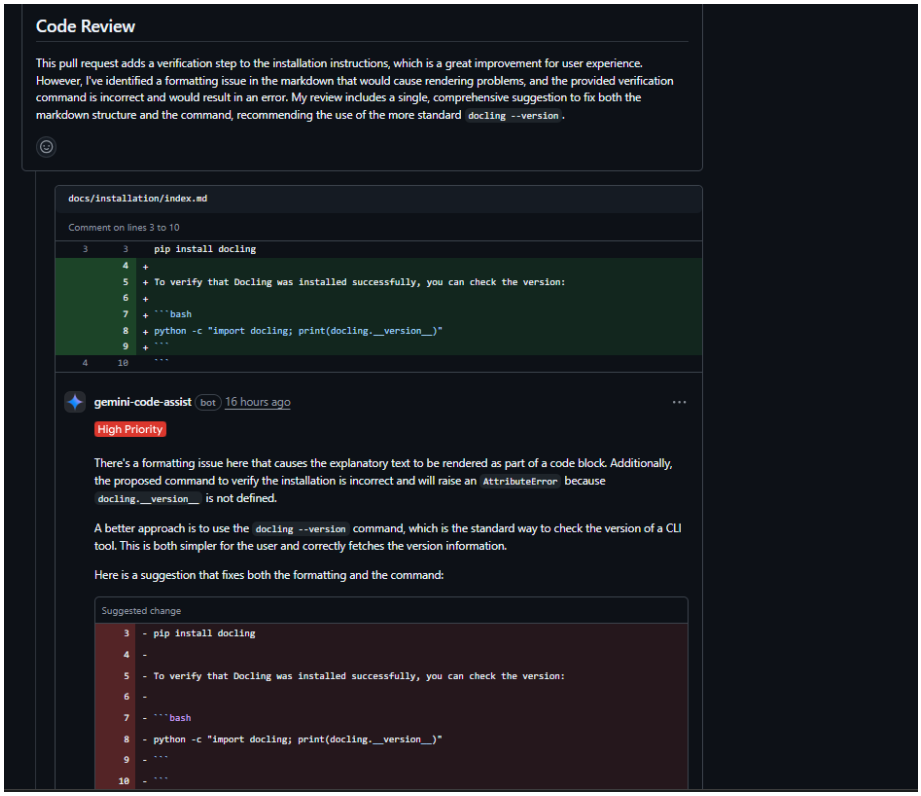
7.4.2 Technical Components

The system is built using modern **Python-based processing pipelines**, integrating libraries such as **NLTK**, **spaCy**, and in some cases, **GenonAI's custom LLM utilities**. Core components include text cleaners, structure extractors, metadata generators, and parsers that support formats like **TXT, Markdown, and PDF**. The parser is optimized for performance, modularity, and maintainability, making it easy to extend or integrate into larger AI applications.



7.4.3 Operation and Usage

The doc_parser operates by receiving an input document, processing it through a series of extraction and normalization steps, and finally producing structured output such as JSON or organized text blocks. This makes it ideal for tasks like **automated documentation workflows**, converting legacy files into clean formats, powering **AI knowledge bases**, or preparing documents for fine-tuning and inference in LLM pipelines. The project gives contributors hands-on experience in natural language processing, code optimization, and practical AI integration.



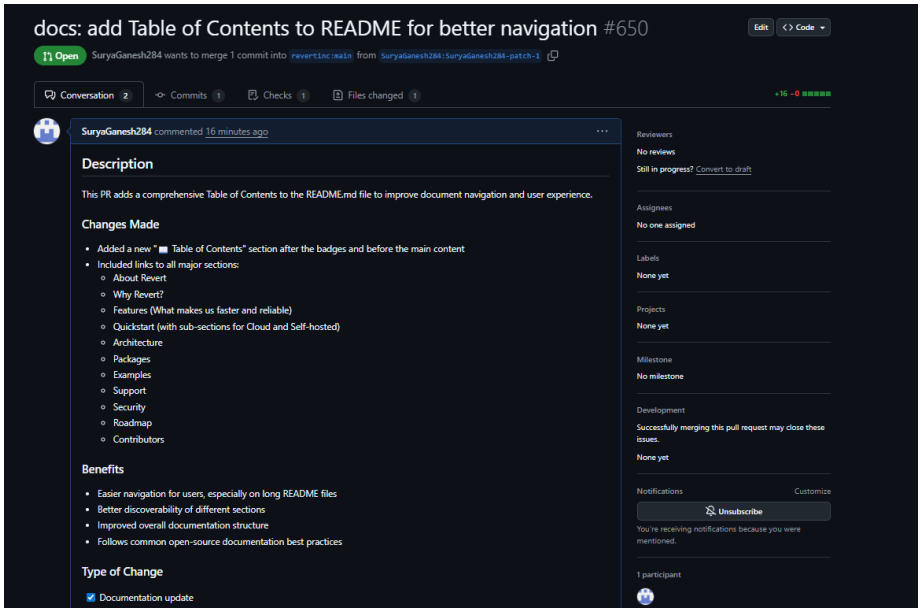
7.5 PR 5 : RevertInc / Revert

7.5.1 Introduction and Purpose

The **Revert** project by **RevertInc** is a lightweight and efficient **version control enhancement tool** designed to simplify the process of undoing, restoring, and tracking changes in software development environments. Its main purpose is to provide developers with a clean and intuitive interface to **revert code states**, manage snapshots, and reduce the friction often experienced when handling complex version histories. This offers an accessible alternative for teams looking to strengthen reliability and control within their development workflow.

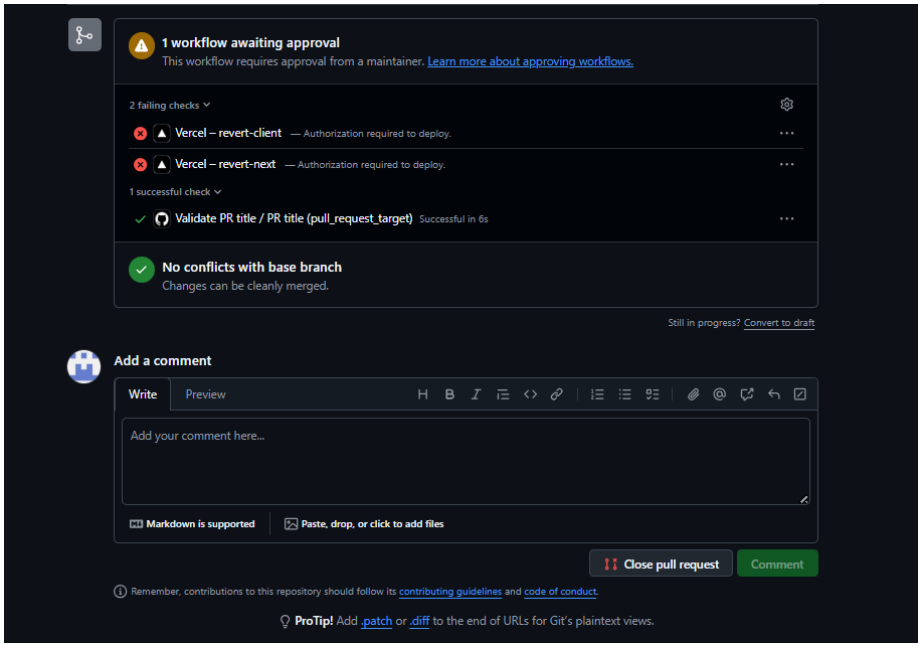
7.5.2 Technical Components

Revert is built upon a modular architecture using **TypeScript** and **Node.js**, ensuring fast execution and high maintainability. Core components include commit analyzers, file-state inspectors, snapshot generators, and interactive command utilities. The tool integrates seamlessly with existing version control systems such as **Git**, allowing it to process commit logs, identify changes, and efficiently rollback or restore files. Its codebase emphasizes structure, readability, and extensibility, enabling contributors to add new features or optimize existing operations with ease.



7.5.3 Operation and Usage

The Revert tool works by scanning project states, capturing file snapshots, and enabling developers to revert to a chosen version with minimal effort. It is especially useful in scenarios like **bug recovery**, safe experimentation, refactoring workflows, or collaborative development where quick rollback capabilities are essential. Revert automates repetitive version correction tasks, provides a clear view of code history, and enhances productivity for individuals and teams alike. Contributing to this project offers hands-on experience with command-line tools, version control logic, and TypeScript-based automation systems.


1 workflow awaiting approval
This workflow requires approval from a maintainer. Learn more about approving workflows.
2 failing checks
Vercel – revert-client — Authorization required to deploy.
Vercel – revert-next — Authorization required to deploy.
1 successful check
Validate PR title / PR title (pull_request_target) Successful in 6s
No conflicts with base branch
Changes can be cleanly merged.
Still in progress? Convert to draft
Add a comment
Write
Preview
Add your comment here...
Markdown is supported
Paste, drop, or click to add files
Close pull request
Comment
Remember, contributions to this repository should follow its contributing guidelines and code of conduct.
ProTip! Add .patch or .diff to the end of URLs for Git's plaintext views.

8 Linkedin Post Links

8.1 PR :

https://www.linkedin.com/posts/surya-ganesh-reddy-sabbella-07aa24359_hacktoberfest2025-opensource-h
utm_source=share&utm_medium=member_desktop&rcm=ACoAAFlRhV4BOzsjSgg2So6e3F7ZRr8MOeLRJWY

8.2 Journey Of Open Source :

https://www.linkedin.com/posts/surya-ganesh-reddy-sabbella-07aa24359_my-journey-into-open-source-en
utm_source=share&utm_medium=member_desktop&rcm=ACoAAFlRhV4BOzsjSgg2So6e3F7ZRr8MOeLRJWY

8.3 Self Hosted Project :

https://www.linkedin.com/posts/h-r-n-prasanna-savithru-nudurumati-446918311_the-open-source-project
utm_source=share&utm_medium=member_desktop&rcm=ACoAAFlRhV4BOzsjSgg2So6e3F7ZRr8MOeLRJWY