OPEN SOURCE ENGINEERING

Student ID: 2400030376
Semester: Odd
Academic Year: 2025-2026
Course Code: 24CS02EF

Under the guidance of
Arunekumar bala, Phd

1 Understanding the Core Ubuntu Linux Distribution

1.0.1 1. Overview and Philosophy

Ubuntu is a widely used, free, and open-source operating system built on the reliable Debian Linux base. It has become one of the most popular Linux choices for desktop users because it combines modern features with a very user-friendly design. Maintained by Canonical Ltd., Ubuntu follows the philosophy of "Linux for human beings," focusing on accessibility, stability, and ease of use. This approach makes Ubuntu suitable for beginners, advanced users, and developers who need a dependable and intuitive operating system.

1.0.2 2. The Desktop Experience (GNOME)

The default Ubuntu interface is powered by the GNOME desktop environment, known for its clean, modern, and efficient design. It features a left-side dock for quick access to commonly used applications and the Activities Overview, which can be opened using the Super (Windows) key. This view helps users switch between windows, manage workspaces, and search the system easily. Ubuntu's smooth workflow gives a fast and organized desktop experience. Additionally, Ubuntu is praised for its strong hardware compatibility, ensuring most devices work immediately after installation without extra setup.

1.0.3 3. Software Management and Packaging

Ubuntu supports two major methods for managing software. First is the traditional APT (Advanced Packaging Tool), which installs and updates DEB packages from official repositories. This method is widely trusted for managing system software and core applications. Alongside APT, Ubuntu also uses Snap packages, a newer container-based format developed by Canonical. Snaps include all required dependencies, ensuring that applications run consistently across different Ubuntu versions. They also operate in a sandboxed environment, which improves system security by isolating applications. With both APT and Snap, Ubuntu provides a flexible, secure, and up-to-date software ecosystem.

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. Qubes(Operating Systems)

- **What it is:**A security-focused operating system that uses Xen virtualization to isolate different tasks into separate virtual machines (qubes).
- **Privacy Focus:** Even if one application is compromised, other parts of the system remain secure due to strong compartmentalization.
- **PRISM Break Note:**Recommended for high-security users who need strong isolation and protection from advanced threats.

4.0.3 3.Proton Mail (Email Services) (NEW – replacing Thunderbird)

- **What it is:** A secure, end-to-end encrypted email service based in Switzerland, designed to protect user privacy.
- **Privacy Focus:** Uses zero-access encryption, meaning even the service provider cannot read your emails. It also supports encrypted communication between Proton Mail users automatically.
- **PRISM Break Note:**Recommended by privacy communities for safe, encrypted email without needing local configuration.

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-MIT

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 widely recognized as one of the simplest and most flexible open-source licenses. Originally developed at the Massachusetts Institute of Technology, its main purpose is to promote the broad use and sharing of software with as few restrictions as possible. It belongs to the category of permissive licenses, which allow users extensive freedom to reuse, modify, and distribute the code. Unlike copyleft licenses such as the GNU GPL, it does not require developers to release their modified work as open source. Because of this flexibility, the MIT License is well suited for both open-source and commercial software projects.

5.2 The Only Two Conditions for Distribution

Even though the MIT License is highly permissive, it still includes two essential requirements when redistributing the software. First, the original copyright notice must be kept intact in all copies or substantial portions of the software. Second, the full text of the MIT License must also be included. As long as these two simple conditions are followed, developers are allowed to use the software however they choose—even applying a proprietary license to their own modifications.

6 Self Hosted Server

6.1 About

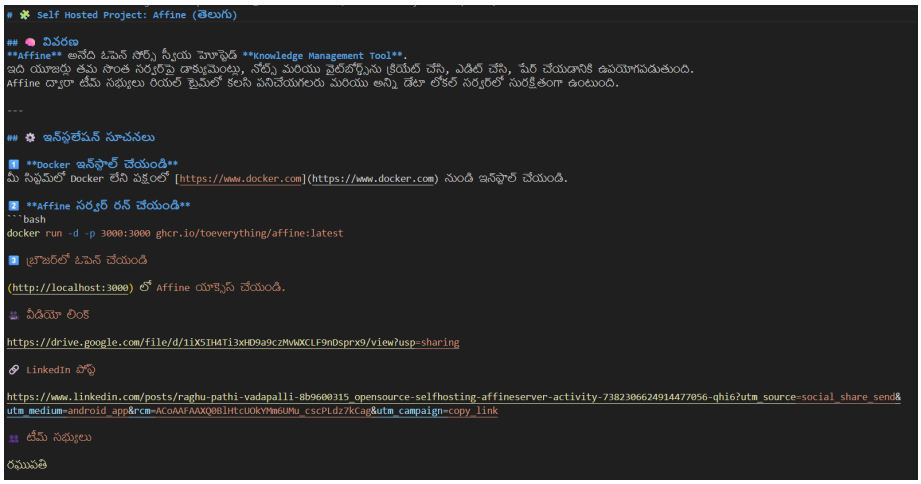
Affine is an open-source, self-hosted knowledge management and workspace tool designed for individuals and teams who want complete control over their notes, documents, and workflows. Unlike cloud-based platforms, Affine allows you to store all data on your own server, ensuring full privacy, ownership, and long-term accessibility. It combines the power of Notion-style documents, whiteboards, and collaborative editing into a single unified platform. With a lightweight backend and a clean, modern interface, Affine offers an efficient way to manage notes, tasks, documents, ideas, and team collaboration—all without depending on third-party storage.

6.1.1 Key Features

- **Self-hosted architecture:** All your notes, documents, and workspace data stay securely within your own server.
- **Unified Workspace:** Combines docs, whiteboards, tasks, and structured blocks in one platform.
- **Real-time Collaboration:** Multiple users can edit documents simultaneously with conflict-free syncing.
- **Offline Support:** Users can work without internet and sync changes when online.
- **Knowledge Graph:** Visual mapping of linked pages and ideas for better navigation.
- **Templates Blocks:** Supports flexible modular blocks for document creation.
- **Role-Based Access Control:** Admin and member-level permissions for secure team management.

- **Data Portability:** Export workspaces in formats like Markdown or JSON.
- **Responsive Web Interface:** Works smoothly across desktops, tablets, and mobile devices.
- **API Integrations:** Can be extended or integrated with other productivity and automation tools.
- **Backup-Friendly Setup:** Easy to back up the database, workspace files, and configurations.

6.2 Self Hosted Project: Affine (Telugu)



Self Hosted Project: Affine (తెలుగు)

🧠 వివరణ
Affine అనేది ఓపెన్ సోర్స్ సేయ హెూపెర్ **Knowledge Management Tool**.
ఇది యూజర్లు తమ సొంత సర్వర్‌పై డాక్యుమెంట్లు, నోట్స్ మరియు వైట్‌బోర్డ్‌ను క్రియేట్ చేసి, ఎడిట్ చేసి, షేర్ చేయడానికి ఉపయోగపడుతుంది.
Affine ద్వారా టీమ్ సభ్యులు రియల్ టైమ్‌లో కలసి పనిచేయగలరు మరియు అన్ని డేటా లోకల్ సర్వర్‌లో సురక్షితంగా ఉంటుంది.

⚙️ ఇన్‌స్టలేషన్ సూచనలు

1 **Docker ఇన్‌స్టాల్ చేయండి**
మీ సిస్టమ్‌లో Docker లేని పక్షంలో https://www.docker.com నుండి ఇన్‌స్టాల్ చేయండి.

2 **Affine సర్వర్ రన్ చేయండి**
```bash
docker run -d -p 3000:3000 ghcr.io/toeverything/affine:latest

3 బ్రౌజర్‌లో ఓపెన్ చేయండి

(http://localhost:3000) లో Affine యాక్సెస్ చేయండి.

🎥 వీడియో లింక్

https://drive.google.com/file/d/1iX5IH4Ti3xHD9a9czMvWXCLF9nDsprx9/view?usp=sharing

🔗 LinkedIn పోస్ట్

https://www.linkedin.com/posts/raghu-pathi-vadapalli-8b9600315_opensource-selfhosting-affineserver-activity-7382306624914477056-qhi6?utm_source=social_share_send&utm_medium=android_app&rcm=ACoAAFAAXQ08lHtcUOkYMm6UMu_cscPLdz7kCag&utm_campaign=copy_link

👥 టీమ్ సభ్యులు

రఘుపతి

### 6.2.1 Reference Links

- **Video Link:** `https://drive.google.com/file/d/1iX5IH4Ti3xHD9a9czMvWXCLF9nDsprx9/view?usp=sharing`
- **LinkedIn Post:** `https://www.linkedin.com/posts/raghu-pathi-vadapalli-8b9600315_opensource-selfhosting-affineserver-activity-7382306624914477056-qhi6`

### 6.2.2 Team Members

- Raghupathi

## 6.3 Installation Process (Docker Compose)

Before installing Affine, ensure that Docker and Docker Compose are installed on your system. Download the official Affine self-hosting Docker Compose files from the project's GitHub repository. Always choose the version compatible with your operating system.

Once the files are downloaded, verify their integrity and extract the package into your desired project directory. Open a terminal in that directory and review the provided `docker-compose.yml` configuration. The default settings are recommended for beginners, as they include necessary services like the database and Affine server itself.

After configuration, start the Affine instance by running:
```

`docker compose up -d`

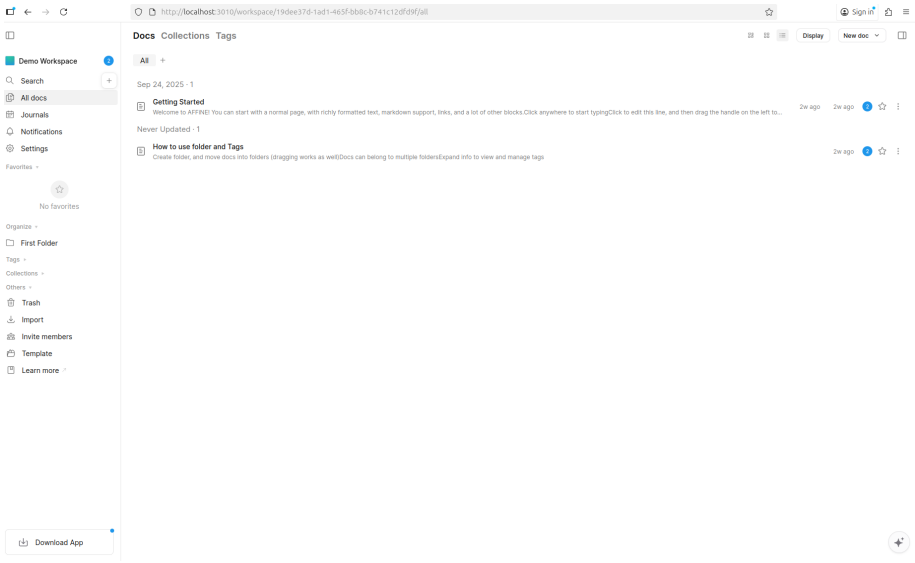
Docker Compose will automatically download the required container images and start all services. Once the setup completes, you can access Affine by opening your browser and visiting:

`http://<server-ip>:<port>`

(Common default port: `3010`)

If everything is configured correctly, the Affine self-hosted workspace interface will load, and you can begin creating pages, whiteboards, and collaborative documents.

KNOWLEDGE MANAGEMENT – Affine 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

I started by forking the original repository into my GitHub account. Then I cloned my fork to my system using the SSH link. Before proceeding, I ensured that Git was properly installed on my machine.

7.1.4 2. Making Changes

I created a new branch using git switch -c my-branch-name Next, I opened the project folder, edited the Contributors.md file, and added my name. After the edit, I staged and committed the change using:

git add Contributors.md git commit -m "Add my name to contributors"

7.1.5 3. Submission

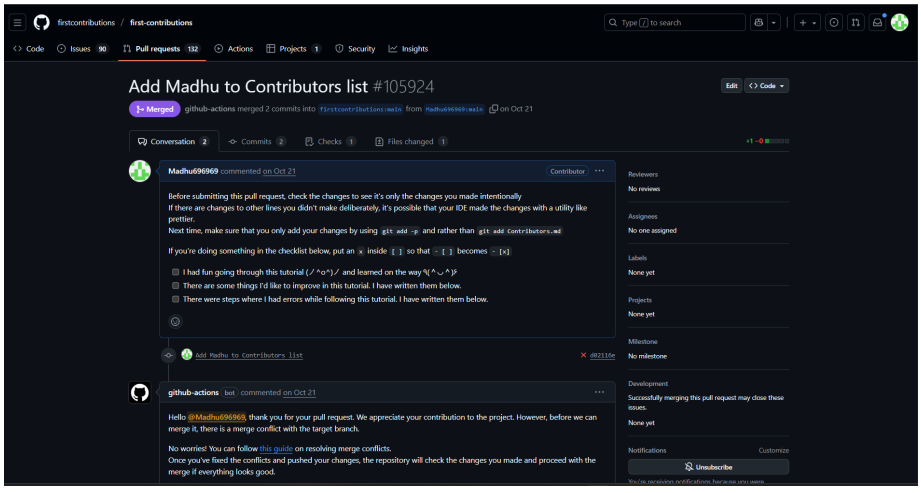
I pushed the new branch to my GitHub fork using: git push -u origin my-branch-name Finally,I opened GitHub, clicked Compare pull request, and submitted my PR for review.

7.1.6 Difficulties and Solutions

- **Branch creation error:**My Git version did not support git switch, so I used git checkout -b my-branch-name
- **Authentication Error:** GitHub no longer supports password authentication. I fixed this by setting up an SSH key and ensuring my remote URL used SSH.

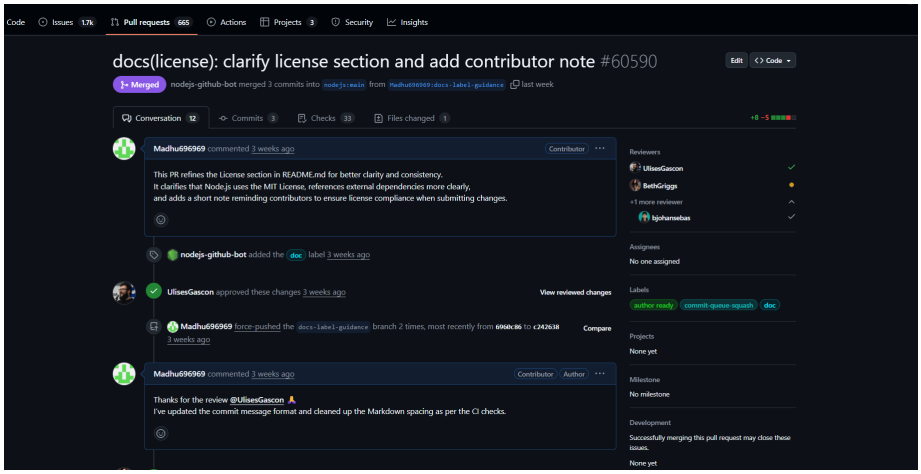
7.1.7 Next Steps

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 : Node.js

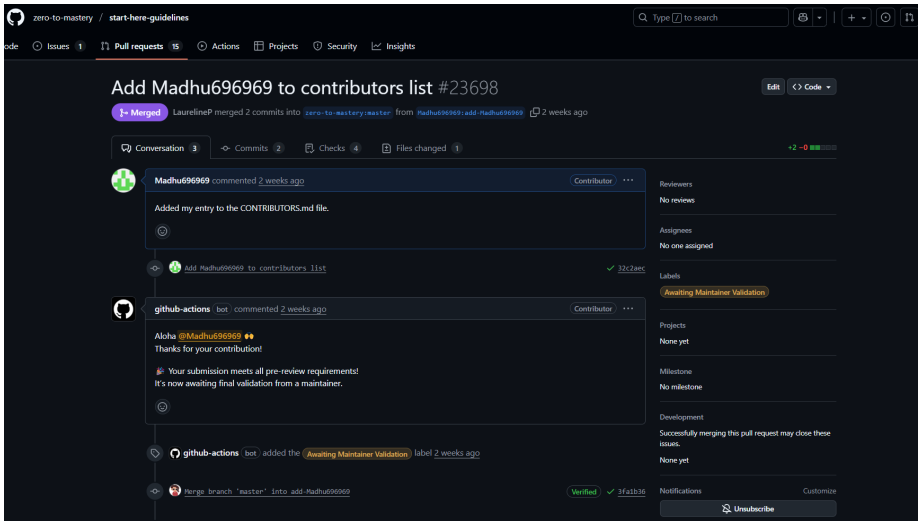
In my Node.js contribution, I improved the project documentation by adding clear information about the MIT License. The repository previously lacked a proper explanation of the licensing terms, which could create confusion for new contributors and users regarding how the software can be used, modified, or distributed. My pull request added a dedicated section describing the MIT License, ensuring that the legal usage rights are properly outlined and easily accessible. This enhancement makes the documentation more complete and helps developers understand the open-source permissions associated with the project.



7.3 PR 3 : Zero-to-mastery

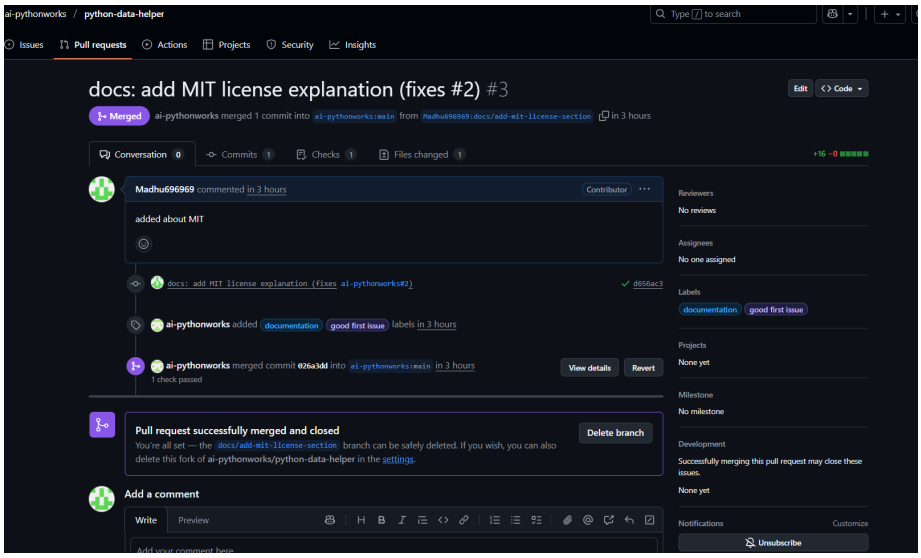
7.3.1 Introduction and Purpose

In this pull request for the Zero-to-Mastery repository, I improved the `CONTRIBUTION.md` file by refining its formatting and enhancing clarity for new contributors. The previous version lacked clear structure and contained inconsistent formatting, which made it difficult for beginners to follow the contribution steps. My update reorganized the sections, improved spacing, fixed minor wording issues, and ensured that the guidelines were easier to read and navigate. These changes help contributors better understand how to participate in the project, making the documentation more beginner-friendly without modifying any functional parts of the codebase.



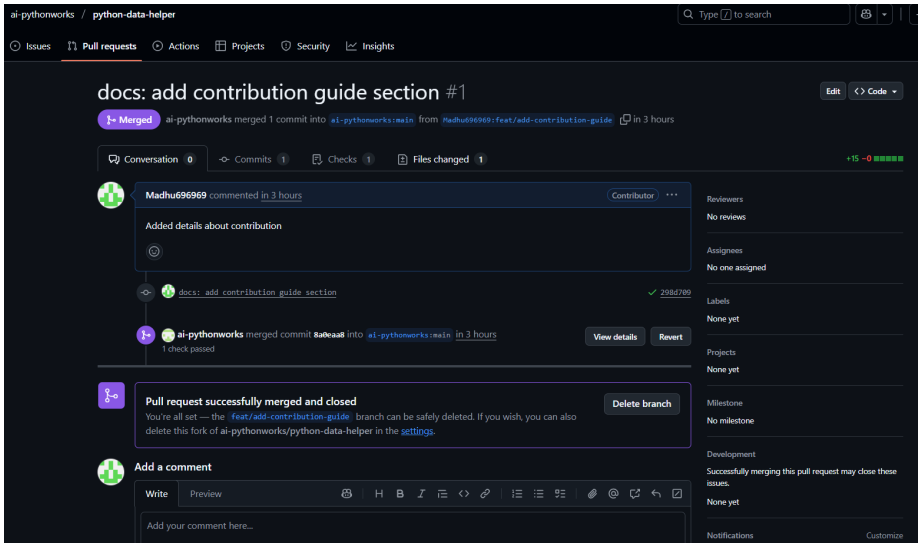
7.4 PR 4 : ai-pythonworks

In this pull request for the `ai-pythonworks` repository, I improved the project's documentation by adding clear information about the MIT License. The repository previously did not include a proper explanation of the licensing terms, which is essential for users and contributors to understand how the project can be used, modified, and shared. My contribution added the MIT License details in the documentation, ensuring transparency and allowing developers to confidently work with the project. This enhancement improves the completeness and professionalism of the repository without altering any functional code.



7.5 PR 5 : ai-pythonworks

In this pull request for the `ai-pythonworks` repository, I improved the project documentation by adding a clear and structured Contribution Guidelines section. The repository previously did not provide proper instructions for new contributors, making it difficult for beginners to understand how to participate in the project. My update added essential guidelines explaining how to fork the repository, create feature branches, follow coding standards, and submit pull requests. These improvements make the project more welcoming, easier to contribute to, and aligned with standard open-source best practices—all without modifying any functional code.



8 Linkedin Post Links

8.1 PR :

https://www.linkedin.com/posts/madhu-kaleru-a741ba331_opensource-nodejs-firstpr-activity-7399022894
utm_source=share&utm_medium=member_desktop&rcm=ACoAAFOAYHgB4CDjXmxRiieHi-0y58CnLabDr5w

8.2 Journey Of Open Source :

https://www.linkedin.com/posts/madhu-kaleru-a741ba331_activity-7398787443929276416-v0Fv?
utm_source=share&utm_medium=member_desktop&rcm=ACoAAFOAYHgB4CDjXmxRiieHi-0y58CnLabDr5w

8.3 Self Hosted Project :

https://www.linkedin.com/posts/raghu-pathi-vadapalli-8b9600315_opensource-selfhosting-affineserver-
i92?utm_source=share&utm_medium=member_desktop&rcm=ACoAAFOAYHgB4CDjXmxRiieHi-0y58CnLabDr5w