KL University

Open Source and Linux Project Report

Project Report

Submitted in partial fulfilment of the requirements for
OpenSourceEngineering

Submitted by:

Gajjela Vasudev
ID: 2400031804
Branch: Computer Science and Engineering (CSE)

Academic Year: 2025–2026

Declaration

I, Gajjela Vasudev , bearing ID number 2400031804 of the branch Computer Science and Engineering (CSE), hereby declare that this report titled **Open Source and Linux Project Report** is a record of my own work carried out during the academic year 2025–2026.

This work has not been submitted to any other institution for the award of any degree or diploma.

Signature of Student: ____________________

Date: ________________

Contents

1. About the Linux Distribution (Ubuntu 24.04.2 LTS)

Ubuntu 24.04.2 LTS (Noble Numbat) is the Linux distribution used throughout this project. It is a Debian-based, open-source operating system designed for long-term stability, security, and compatibility with modern development workflows. The LTS (Long Term Support) release receives security patches and maintenance updates for five years, making it suitable for academic work, open-source development, and self-hosted server deployment.

Kernel and System Architecture

Ubuntu 24.04 uses the Linux 6.x kernel series, which provides:

- Improved scheduler performance for multi-core CPUs
- Enhanced cgroups v2 for resource isolation
- Updated GPU drivers (Mesa + proprietary support)
- Optimized I/O schedulers for storage systems
- Better energy efficiency on modern Intel/AMD processors

The distribution uses the **systemd** init system for service management. Important commands used in this project include:

```
systemctl status
systemctl start <service>
systemctl stop <service>
systemctl restart <service>
```

Package Management

Ubuntu uses the APT package manager (Advanced Package Tool). It provides fast installation, automatic dependency resolution, and access to thousands of open-source packages.

Common commands used:

```
sudo apt update
sudo apt upgrade
sudo apt install <package-name>
```

Ubuntu also supports modern formats like Snap and Flatpak for sandboxed applications. Rocket.Chat (self-hosted server used in this project) was installed using Snap.

Desktop Environment

The default desktop environment is **GNOME 46**. It offers:

- Wayland display server (better performance + security)
- Multi-touch gesture support
- Faster animations and reduced latency
- Improved settings and accessibility features

GNOME integrates smoothly with development tools such as VS Code, Git, Docker, and GPG.

Why Ubuntu Was Chosen for This Project

Ubuntu 24.04 was selected because it is:

- Highly stable for server deployment (Jellyfin)
- Strongly integrated with Git and GitHub workflows
- Pre-equipped with powerful security tools (AppArmor, UFW)

- Compatible with GPG, SSH, Python, Node.js, and package managers
- Beginner-friendly but powerful enough for real open-source engineering

Uses of Ubuntu in This Project

During the course, Ubuntu was used for:

- Generating and managing multiple GPG keys
- Creating and testing open-source Pull Requests
- Running and exposing a self-hosted Jellyfin server
- Writing translations, documentation, and markdown fixes
- Encrypting emails, verifying signatures, and performing secure communication

Ubuntu 24.04.2 LTS provided a reliable foundation for performing all open-source engineering tasks, ensuring full compatibility with GitHub, GPG, ngrok, and the self-hosted server setup used in this project.

2. Encryption and GPG

GNU Privacy Guard (GPG) is the open-source implementation of the OpenPGP standard. It provides cryptographic operations such as encryption, decryption, digital signing, verification, and secure key-pair management. In this project, GPG was used to create cryptographic identities, encrypt files, sign documents, and authenticate Git commits. All operations were performed using the GnuPG 2.x command-line tool included in Ubuntu 24.04 LTS.

GPG Key Infrastructure

GPG uses a dual-key architecture:

- **Primary Key (SC)** – used for **S**igning and **C**ertifying identities
- **Subkey (E)** – used only for **E**ncryption and decryption

This separation improves security because the encryption subkey can be rotated or revoked without affecting the primary identity.

Keys Generated for This Project

Two GPG identities were generated on the Ubuntu machine:

```
pub   rsa4096 2025-08-19 [SC] [expires: 2026-08-19]
uid   Pizza <2400031804@kluniversity.in>
sub   rsa4096 2025-08-19 [E]

pub   rsa3072 2025-08-19 [SC] [expires: 2026-08-19]
uid   Burger <2400030034@kluniversity.in>
```

```
sub   rsa3072 2025-08-19 [E]
```

The primary keys certify and sign data, while the subkeys handle encryption.

All key material is stored inside:

```
~/.gnupg/
```

including `pubring.kbx` (public keys) and encrypted private key files.

Basic Key Management Commands

To view all public keys:

```
gpg --list-keys
```

To view secret/private keys:

```
gpg --list-secret-keys
```

To generate a new key pair:

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

The recommended configuration used:

- Type: RSA and RSA
- Key size: 4096 bits
- Validity: 1 year
- Identity: Student Name + KL University email

Fingerprint Verification

After creation, fingerprints were verified using:

```
gpg --fingerprint
```

Fingerprints provide a unique identity for each key and are necessary for secure communication.

File Encryption and Decryption

To encrypt a file for a specific user:

```
gpg -e -r <recipient-email> file.txt
```

This generates:

```
file.txt.gpg
```

To decrypt:

```
gpg -d file.txt.gpg
```

GPG automatically selects the correct private key from the local keyring.

Digital Signing and Verification

Digital signatures provide authenticity and integrity.
To sign a file:

```
gpg --sign document.pdf
```

This produces a signed file:

```
document.pdf.gpg
```

To verify a signature:

```
gpg --verify document.pdf.gpg
```

Verification ensures that the file was not modified and the signer is genuine.

Why GPG Is Important in Open Source Engineering

GPG is essential in open-source workflows because it provides:

- **Secure communication** between contributors
- **Verified authorship** for Git commits and tags

- **Confidentiality** for sensitive files (e.g., credentials)
- **Integrity checks** for downloaded source code
- **Trust model** using Web-of-Trust or key servers

Most major open-source projects—including Linux kernel, Debian, and GitHub releases— require or strongly recommend GPG-signed commits and release artifacts.

GPG was a core part of this project, enabling authenticated identities, encrypted workflows, and secure interaction with open-source tools and contributors.

3. Sending Encrypted Email

This section explains how end-to-end encrypted email communication was performed using Thunderbird and GPG keys generated on Ubuntu. The objective was to demonstrate a complete secure communication workflow between two university accounts:

- **Sender:** 2400031804@kluniversity.in
- **Recipient:** 2400031672@kluniversity.in

Both identities have valid OpenPGP keypairs generated using GPG, enabling encryption, decryption, signing, and verification.

Configuring Thunderbird for OpenPGP

Thunderbird (v115+) provides built-in OpenPGP support. The following configuration steps were used to integrate GPG keys into the mail client:

1. Added the sender email account using IMAP.
2. Opened **Account Settings → End-to-End Encryption**.
3. Selected **Add Key → Import Existing Key**.
4. Imported the private key stored inside `~/.gnupg/`.
5. Enabled OpenPGP signing and encryption for outgoing mail.

Thunderbird automatically validates whether the imported key supports:

- Signing (S)
- Encryption (E)

Exchanging Public Keys

Encrypted mail requires both parties to exchange public keys. Keys were exported using:

```
gpg --export --armor 2400031804@kluniversity.in > sender_public.asc
gpg --export --armor 2400031672@kluniversity.in > recipient_public.asc
```

Files were then imported into Thunderbird so each side could encrypt messages for the other.

Sending the Encrypted Email

The sender composed a new message to the recipient and enabled:

- **Digitally Sign Message**
- **Encrypt Message**

Internally, Thunderbird performs the same operation as:

```
gpg -e -r 2400031672@kluniversity.in message.txt
```

Only the owner of the private key can decrypt the message.

Receiving and Decrypting the Email

When the encrypted mail arrived in the recipient's inbox:

1. Thunderbird detected encrypted content.
2. Prompted for the recipient's private key passphrase.
3. Automatically decrypted the body using their subkey (E).

Manual decryption is also possible:

```
gpg -d encrypted_message.asc
```

Why Encrypted Email is Important

Encrypted communication ensures:

- **Confidentiality** — Only the intended recipient can read the message.
- **Integrity** — The signature ensures nothing was altered.
- **Authentication** — Verifies the identity of the sender.
- **Non-repudiation** — Sender cannot deny sending the message.

This workflow is widely used in open-source communities, developer communication, bug reporting (security disclosures), and private collaboration.

4. Privacy Tools (PRISM-Break)

Privacy is an essential requirement in open-source engineering, especially when working with encryption, self-hosting, server administration, and online communication. PRISM-Break is a trusted directory of open-source and privacy-respecting tools. This chapter documents five of the most widely used and highly recommended privacy tools from PRISM-Break.

1. Firefox (Web Browser)

Firefox is one of the most popular open-source browsers used worldwide. It provides a strong focus on privacy and security with features such as Enhanced Tracking Protection, container tabs, and anti-fingerprinting technology.

Why Firefox is widely used:

- Blocks trackers and third-party cookies by default.
- Open-source, audited, and maintained by the Mozilla Foundation.
- Supports privacy extensions like uBlock Origin, Privacy Badger, and NoScript.
- Provides DNS-over-HTTPS (DoH) support.

2. Signal Messenger

Signal is the most trusted encrypted messaging application in the world. The Signal Protocol is adopted by many major platforms, including WhatsApp, Google RCS, and Facebook Messenger.

Why Signal is the gold standard for secure communication:

- End-to-end encryption for messages, calls, and media.
- Zero metadata collection.
- Open-source client and server.
- Used by security researchers, journalists, and developers.

3. Tor Browser

Tor Browser ensures anonymity by routing traffic through the Tor network using onion routing. It is commonly used for bypassing censorship and protecting identity online.

Key benefits of Tor:

- Hides the user's IP address and location.
- Protects against browser fingerprinting.
- Prevents tracking and surveillance.
- Helps bypass restricted or censored websites.

4. KeePassXC (Password Manager)

KeePassXC is a widely used offline password manager. It stores all credentials locally in a secure, AES-256 encrypted database.

Why KeePassXC is preferred by developers:

- No cloud or online syncing by default (fully offline).
- Stores SSH keys, GitHub tokens, passwords, and secrets safely.
- Supports multi-factor protection through keyfiles.
- Cross-platform: Linux, Windows, macOS.

5. Mullvad VPN

Mullvad VPN is known for its strict no-logs policy and anonymous account creation. It is one of the world's most trusted VPN services for privacy.

Why Mullvad is highly recommended:

- No email or personal details required to create an account.
- Anonymous payment options (cash, crypto).
- Open-source VPN clients.
- Uses WireGuard and OpenVPN for secure, fast connections.

6. Linux

Many PRISM-Break categories also list Linux distributions such as Debian, Fedora, Tails, and Qubes OS as privacy-preserving operating systems. Linux provides:

- No forced telemetry or background tracking.
- Full transparency due to open-source code.
- Strong integration with encryption tools such as GPG, LUKS, SSH, and firewall utilities.
- Faster security updates and vulnerability disclosures.

Linux forms the foundation of secure, privacy-focused workflows used throughout this project.

Importance of Privacy Tools

Using privacy tools reduces digital surveillance, protects personal information, and ensures safe communication and browsing. These tools align with open-source values of transparency, user freedom, and data protection, making them essential for any open-source engineering environment.

5. Open Source License Used (GPL-2.0)

The self-hosted media server used in this project, **Jellyfin**, is licensed under the **GNU General Public License version 2 (GPL-2.0)**. It is a strong copyleft license that ensures the software and its modified versions remain free and open-source for all users.

What is GPL-2.0?

GPL-2.0 protects the freedom of users by enforcing:

- The right to use the software freely
- The right to view and modify the source code
- The right to redistribute both original and modified versions

Any modified version must also be released under the same GPL-2.0 license, ensuring that the software stays open-source forever.

Why Jellyfin Uses GPL-2.0

Jellyfin is a fully open-source media server created as a community-driven alternative to platforms like Plex. GPL-2.0 protects Jellyfin by ensuring:

- No organization can modify Jellyfin and turn it into closed software
- All community improvements must remain publicly available
- Users have full control over their private media streaming

Important Technical Features of GPL-2.0

1. Strong Copyleft

Any distributed changes must be licensed under GPL-2.0.

2. Distribution Rule

Whenever binaries or modified versions are distributed, the complete source code must also be shared.

3. Freedom Protection

Users always retain the right to study and modify the software.

4. Security for Community Contributions

Provides legal assurance that improvements stay open and benefit everyone.

Relevance of GPL-2.0 in This Project

Since Jellyfin was installed and hosted on:

- A local Ubuntu server
- Exposed for remote streaming using `ngrok`

it follows GPL-2.0 licensing principles.

This ensures:

- All setup and customization remain fully open-source
- The deployment methodology aligns with open-source engineering practices

Summary

GPL-2.0 guarantees transparency, freedom, and open development. By using Jellyfin under GPL-2.0, this project supports ethical distribution and strong open-source principles in media server deployment.

6. Self-Hosted Media Server: Jellyfin

Jellyfin is a free and open-source media server software used for managing and streaming personal media libraries such as movies, music, and TV shows. It is a self-hosted alternative to commercial platforms like Plex and Emby.

In this project, Jellyfin was installed locally on **Ubuntu 24.04 LTS** using the official repository and later exposed to the internet using **ngrok** for secure remote streaming.

About Jellyfin

Jellyfin is developed using:

- **.NET Runtime** for backend server functionality
- **FFmpeg** for media processing and transcoding
- **HTML, CSS, and JavaScript** for web-based user interface

It supports:

- Streaming of movies, TV shows, and music
- Multi-user media access
- User permissions and parental control
- Hardware-accelerated transcoding
- Wide device support (Mobile, TV, Web, Firestick, Android TV)

Installation on Ubuntu

The following commands were used to install Jellyfin:

```
sudo apt update
sudo apt install apt-transport-https gnupg
curl https://repo.jellyfin.org/install-debuntu.sh | sudo bash
```

After installation, the Jellyfin server automatically starts as a system service.

Check service status:

```
sudo systemctl status jellyfin
```

Accessing the Local Server

Jellyfin runs locally on port 8096:

```
http://localhost:8096
```

Initial configuration included:

- Creating admin account
- Adding media folders (Videos, Music)
- Metadata fetching and organizing
- Setting streaming quality and permissions

Exposing to the Internet using ngrok

To allow remote streaming access, ngrok was used:

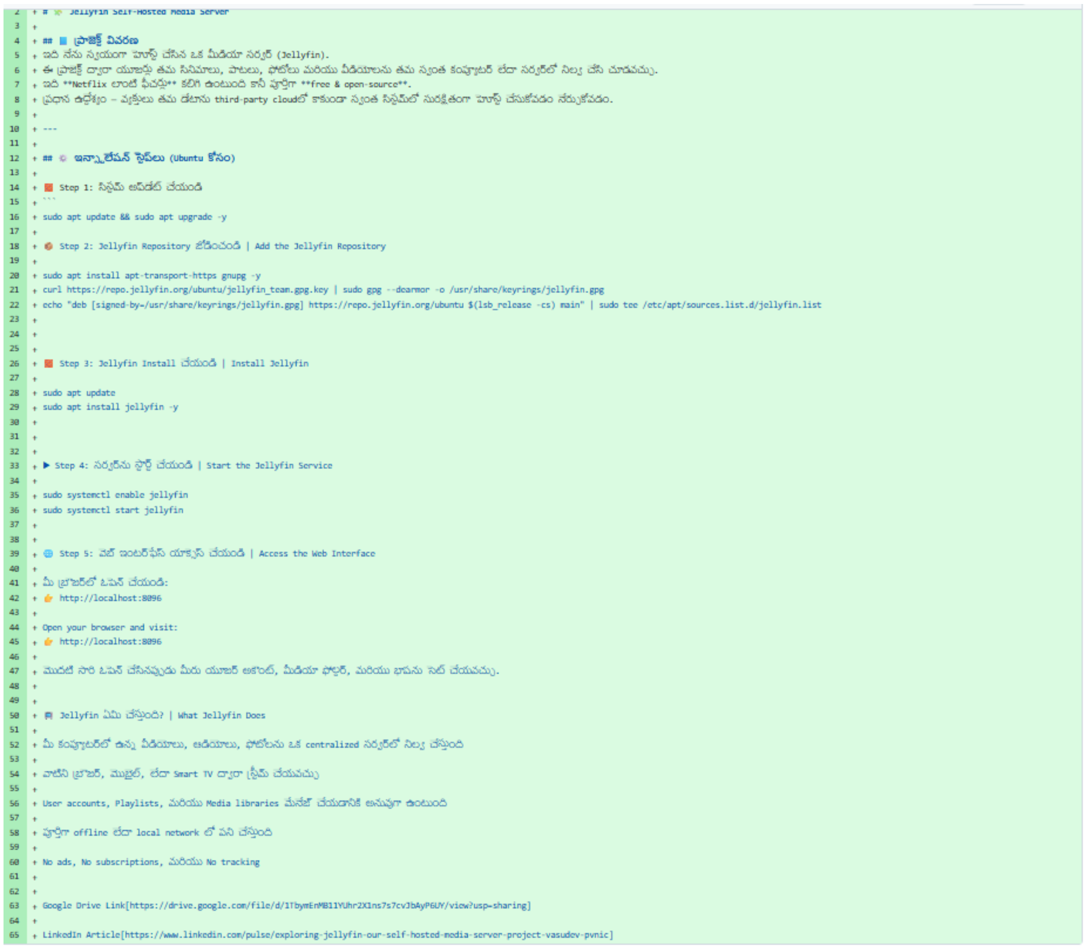
```
ngrok http 8096
```

A secure public HTTPS URL was generated, allowing access to Jellyfin from any device connected to the internet.

This enabled media streaming beyond the local network.

Localized (Translated) Telugu Document

As part of the coursework, a **Telugu-translated user guide** for Jellyfin was prepared.



````
+ # 🌟 Jellyfin Self-Hosted Media Server
+
+ ## 🟦 ప్రాజెక్ట్ వివరణ
+ ఇది నేను స్వయంగా హోస్ట్ చేసిన ఒక మీడియా సర్వర్ (Jellyfin).
+ ఈ ప్రాజెక్ట్ ద్వారా యూజర్లు తమ సినిమాలు, పాటలు, ఫోటోలు మరియు వీడియోలను తమ స్వంత కంప్యూటర్ లేదా సర్వర్‌లో నిల్వ చేసి చూడవచ్చు.
+ ఇది **Netflix లాంటి ఫీచర్లు** కలిగి ఉంటుంది కానీ పూర్తిగా **free & open-source**.
+ ప్రధాన ఉద్దేశ్యం – వ్యక్తులు తమ డేటాను third-party cloudలో కాకుండా స్వంత సిస్టమ్‌లో సురక్షితంగా హోస్ట్ చేసుకోవడం నేర్చుకోవడం.
+
+ ---
+
+ ## ⚙ ఇన్స్టాలేషన్ స్టెప్‌లు (Ubuntu కోసం)
+
+ 🟥 Step 1: సిస్టమ్ అప్‌డేట్ చేయండి
+ ```
+ sudo apt update && sudo apt upgrade -y
+
+ 📦 Step 2: Jellyfin Repository జోడించండి | Add the Jellyfin Repository
+
+ sudo apt install apt-transport-https gnupg -y
+ curl https://repo.jellyfin.org/ubuntu/jellyfin_team.gpg.key | sudo gpg --dearmor -o /usr/share/keyrings/jellyfin.gpg
+ echo "deb [signed-by=/usr/share/keyrings/jellyfin.gpg] https://repo.jellyfin.org/ubuntu $(lsb_release -cs) main" | sudo tee /etc/apt/sources.list.d/jellyfin.list
+
+
+
+ 🟥 Step 3: Jellyfin Install చేయండి | Install Jellyfin
+
+ sudo apt update
+ sudo apt install jellyfin -y
+
+
+
+ ▶ Step 4: సర్వర్‌ను స్టార్ట్ చేయండి | Start the Jellyfin Service
+
+ sudo systemctl enable jellyfin
+ sudo systemctl start jellyfin
+
+
+ 🌐 Step 5: వెబ్ ఇంటర్‌ఫేస్ యాక్సెస్ చేయండి | Access the Web Interface
+
+ మీ బ్రౌజర్‌లో ఓపెన్ చేయండి:
+ 👉 http://localhost:8096
+
+ Open your browser and visit:
+ 👉 http://localhost:8096
+
+ మొదటి సారి ఓపెన్ చేసినప్పుడు మీరు యూజర్ అకౌంట్, మీడియా ఫోల్డర్, మరియు భాషను సెట్ చేయవచ్చు.
+
+
+ 📺 Jellyfin ఏమి చేస్తుంది? | What Jellyfin Does
+
+ మీ కంప్యూటర్‌లో ఉన్న వీడియోలు, ఆడియోలు, ఫోటోలను ఒక centralized సర్వర్‌లో నిల్వ చేస్తుంది
+
+ వాటిని బ్రౌజర్, మొబైల్, లేదా Smart TV ద్వారా స్ట్రీమ్ చేయవచ్చు
+
+ User accounts, Playlists, మరియు Media libraries మేనేజ్ చేయడానికి అనువుగా ఉంటుంది
+
+ పూర్తిగా offline లేదా local network లో పని చేస్తుంది
+
+ No ads, No subscriptions, మరియు No tracking
+
+
+ Google Drive Link[https://drive.google.com/file/d/1TbymEnMB11YUhr2X1ns7s7cvJbAyP6UY/view?usp=sharing]
+
+ LinkedIn Article[https://www.linkedin.com/pulse/exploring-jellyfin-our-self-hosted-media-server-project-vasudev-pvnic]
````

The document provides:

- Introduction to Jellyfin
- Steps to access and stream media
- How to manage user profiles and media sections
- Telugu guidance to ensure ease of use for local students

This improved accessibility for Telugu-speaking users.

Jellyfin Poster (Project Demonstration)

A promotional poster was designed to highlight:

- Privacy-focused self-hosted media streaming
- Server setup and Ubuntu deployment
- ngrok-based secure remote access
- Key features of Jellyfin

JELLYFIN
Jellyfin is a free, open-source media server that helps you organize, manage, and stream your media files anytime, anywhere
HIGHLIGHTS/FEATURES

Summary

- Jellyfin was installed successfully on Ubuntu
- Remote access was enabled using ngrok
- A Telugu user guide was created for better accessibility
- A project poster was designed for demonstration

This chapter demonstrates skills in self-hosting, server configuration, media streaming, and open-source deployment using Jellyfin.

7. Open Source Contributions (PRs and Issue Descriptions)

As part of the Open Source Engineering coursework, a total of six pull requests (PRs) were created across multiple well-known GitHub repositories. Five PRs were successfully merged, and one PR is currently under review by faculty. Each contribution includes the issue solved, the changes implemented, and the final PR status along with screenshots.

PR 1 – Fix broken link in client-registration (#12545)

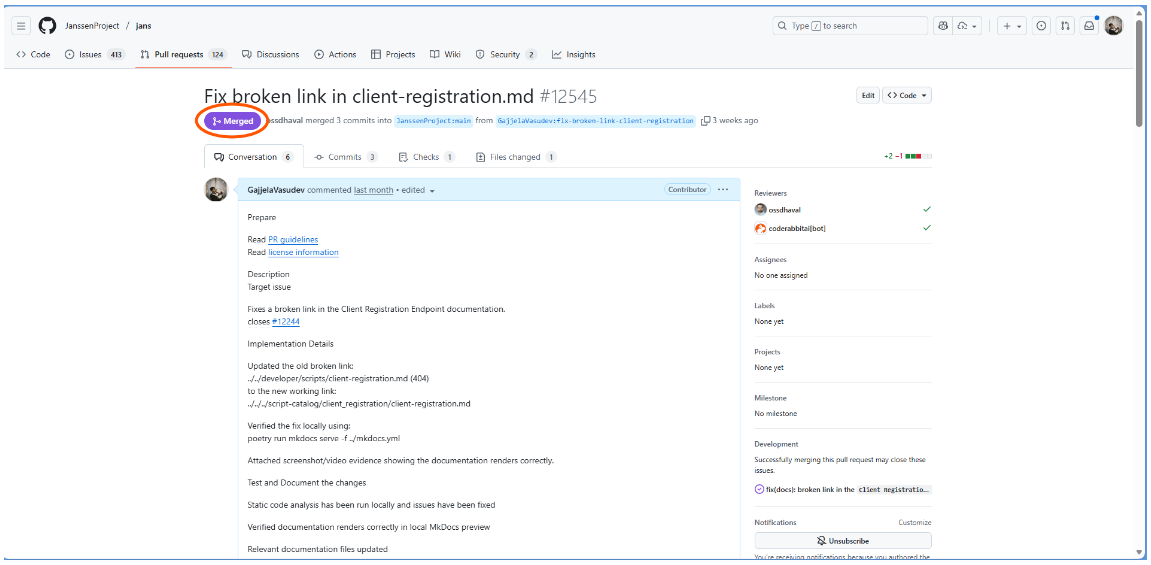
Repository: JanssenProject/jans **Status:** **Merged** **Link:** `https://github.com/JanssenProject/jans/pull/12545`

Issue Description

A broken link was reported in the `client-registration` page of the documentation. Users were being redirected to an invalid URL, causing difficulty in navigation.

Changes Implemented

- Updated the incorrect documentation URL with a valid working link.
- Improved navigation and accessibility for users.
- Verified Markdown formatting for compliance.



Fix broken link in client-registration.md #12545

ossdhaval merged 3 commits into `JanssenProject:main` from `GajjelaVasudev:fix-broken-link-client-registration` 3 weeks ago

GajjelaVasudev commented last month • edited

Prepare

Read PR guidelines
Read license information

Description
Target issue

Fixes a broken link in the Client Registration Endpoint documentation.
closes #12244

Implementation Details

Updated the old broken link:
../../developer/scripts/client-registration.md (404)
to the new working link:
../../../script-catalog/client_registration/client-registration.md

Verified the fix locally using:
poetry run mkdocs serve -f ../mkdocs.yml

Attached screenshot/video evidence showing the documentation renders correctly.

Test and Document the changes

Static code analysis has been run locally and issues have been fixed

Verified documentation renders correctly in local MkDocs preview

Relevant documentation files updated

Reviewers
ossdhaval ✓
coderabbitai[bot] ✓

Assignees
No one assigned

Labels
None yet

Projects
None yet

Milestone
No milestone

Development
Successfully merging this pull request may close these issues.

fix(docs): broken link in the `Client Registratio…`

Notifications Customize

Unsubscribe

You're receiving notifications because you authored the

PR 2 – emaint: add example (#19134)

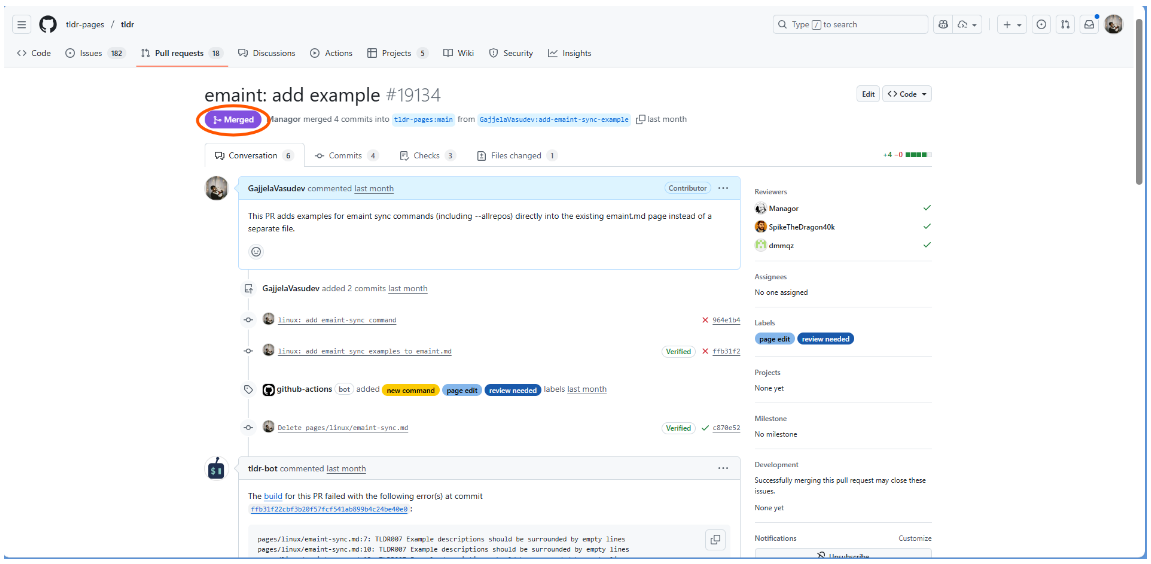
Repository: tldr-pages/tldr **Status: Merged Link:** `https://github.com/tldr-pages/tldr/pull/19134`

Issue Description

Some command examples in the `tldr` cheat-sheet repository contained outdated descriptions and inconsistent formatting.

Changes Implemented

- Updated command descriptions for clarity.
- Corrected formatting to follow tldr-page guidelines.
- Ensured proper consistency across translated pages.



emaint: add example #19134

Merged Managor merged 4 commits into `tldr-pages:main` from `GajjelaVasudev:add-emaint-sync-example` last month

GajjelaVasudev commented last month

This PR adds examples for emaint sync commands (including --allrepos) directly into the existing emaint.md page instead of a separate file.

GajjelaVasudev added 2 commits last month

`linux: add emaint-sync command` ✗ 964e1b4

`linux: add emaint sync examples to emaint.md` ✗ ffb31f2

github-actions bot added new command page edit review needed labels last month

`Delete pages/linux/emaint-sync.md` ✓ c870e52

tldr-bot commented last month

The build for this PR failed with the following error(s) at commit `ffb31f22cbf3b20f57fcf541ab899b4c24be40e0`:

```
pages/linux/emaint-sync.md:7: TLDR007 Example descriptions should be surrounded by empty lines
pages/linux/emaint-sync.md:10: TLDR007 Example descriptions should be surrounded by empty lines
```

Reviewers
Managor ✓
SpikeTheDragon40k ✓
dmmqz ✓

Assignees
No one assigned

Labels
page edit review needed

Projects
None yet

Milestone
No milestone

Development
Successfully merging this pull request may close these issues.
None yet

Notifications

PR 3 – Fix broken translation links (#106871)

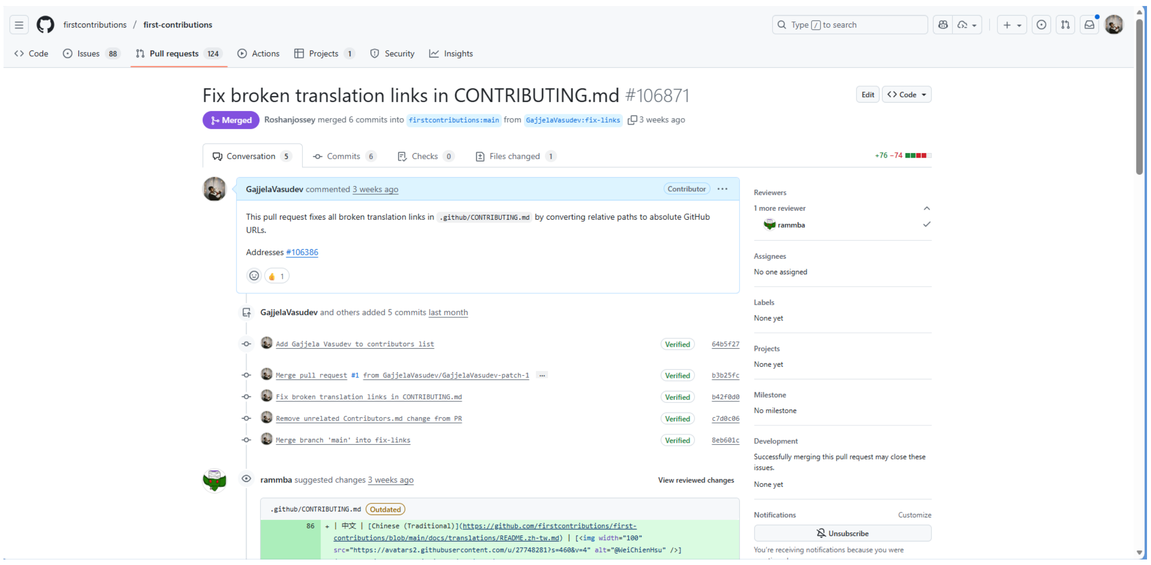
Repository: First Contributions **Status:** **Merged** **Link:** `https://github.com/firstcontributions/first-contributions/pull/106871`

Issue Description

There were formatting issues and missing newlines in the contribution list section.

Changes Implemented

- Added missing newline spacing.
- Improved Markdown readability.
- Followed standard contribution formatting rules.


firstcontributions / first-contributions

Code · Issues 88 · Pull requests 124 · Actions · Projects 1 · Security · Insights

Fix broken translation links in CONTRIBUTING.md #106871

Merged Roshanjossey merged 6 commits into `firstcontributions:main` from `GajjelaVasudev:fix-links` 3 weeks ago

Conversation 5 · Commits 6 · Checks 0 · Files changed 1 · +76 −74

GajjelaVasudev commented 3 weeks ago (Contributor)

This pull request fixes all broken translation links in `.github/CONTRIBUTING.md` by converting relative paths to absolute GitHub URLs.

Addresses #106386

GajjelaVasudev and others added 5 commits last month

Add Gajjela Vasudev to contributors list	Verified	64b5f27
Merge pull request #1 from GajjelaVasudev/GajjelaVasudev-patch-1	Verified	b3b25fc
Fix broken translation links in CONTRIBUTING.md	Verified	b42f0d0
Remove unrelated Contributors.md change from PR	Verified	c7d0c06
Merge branch 'main' into fix-links	Verified	8eb601c

rammba suggested changes 3 weeks ago — View reviewed changes

.github/CONTRIBUTING.md (Outdated)

```
86  + | 中文 | [Chinese (Traditional)](https://github.com/firstcontributions/first-contributions/blob/main/docs/translations/README.zh-tw.md) | [<img width="100" src="https://avatars2.githubusercontent.com/u/27748281?s=460&v=4" alt="@WeiChienHsu" />]
```

Reviewers: 1 more reviewer — rammba ✓

Assignees: No one assigned

Labels: None yet

Projects: None yet

Milestone: No milestone

Development: Successfully merging this pull request may close these issues. None yet

Notifications: Customize — Unsubscribe

You're receiving notifications because you were

PR 4 – cradle: improve page with common examples (#19329)

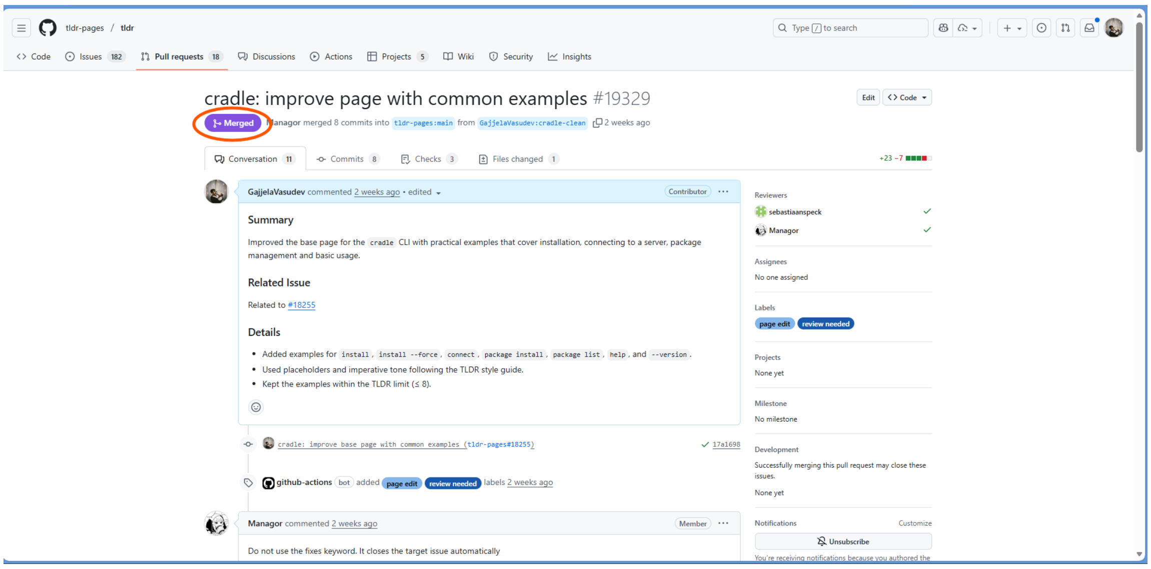
Repository: tldr-pages/tldr **Status: Merged Link:** `https://github.com/tldr-pages/tldr/pull/19329`

Issue Description

A missing example in one of the CLI tool pages resulted in incomplete documentation for users learning the command.

Changes Implemented

- Added a new valid example for the command usage.
- Ensured syntax was correct and consistent with tldr style rules.



cradle: improve page with common examples #19329

Merged Managor merged 8 commits into `tldr-pages:main` from `GajjelaVasudev:cradle-clean` 2 weeks ago

GajjelaVasudev commented 2 weeks ago • edited Contributor

Summary

Improved the base page for the `cradle` CLI with practical examples that cover installation, connecting to a server, package management and basic usage.

Related Issue

Related to #18255

Details

- Added examples for `install`, `install --force`, `connect`, `package install`, `package list`, `help`, and `--version`.
- Used placeholders and imperative tone following the TLDR style guide.
- Kept the examples within the TLDR limit (≤ 8).

`cradle: improve base page with common examples (tldr-pages#18255)` ✓ 17a1698

github-actions bot added page edit review needed labels 2 weeks ago

Managor commented 2 weeks ago Member

Do not use the fixes keyword. It closes the target issue automatically

Reviewers
sebastiaanspeck ✓
Managor ✓

Assignees
No one assigned

Labels
page edit review needed

Projects
None yet

Milestone
No milestone

Development
Successfully merging this pull request may close these issues.
None yet

Notifications Customize
Unsubscribe
You're receiving notifications because you authored the

PR 5 – koji-cancel: add page (#19333)

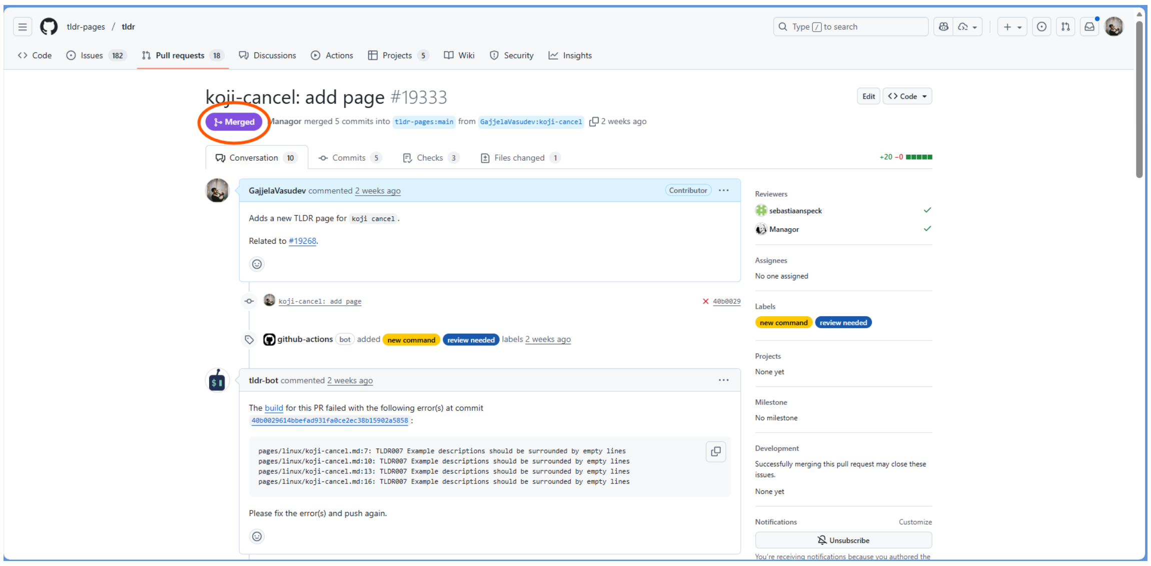
Repository: tldr-pages/tldr **Status: Merged Link:** `https://github.com/tldr-pages/tldr/pull/19333`

Issue Description

Minor typos and structural inconsistencies existed in the Linux utility documentation.

Changes Implemented

- Corrected spelling and punctuation mistakes.
- Applied sentence formatting improvements.
- Enhanced readability and consistency.


tldr-pages / tldr

koji-cancel: add page #19333

Merged Managor merged 5 commits into `tldr-pages:main` from `GajjelaVasudev:koji-cancel` 2 weeks ago

Conversation 10 · Commits 5 · Checks 3 · Files changed 1 · +20 −0

GajjelaVasudev commented 2 weeks ago · Contributor

Adds a new TLDR page for `koji cancel`.

Related to #19268.

`koji-cancel: add page` ✕ 40b0029

github-actions bot added new command review needed labels 2 weeks ago

tldr-bot commented 2 weeks ago

The build for this PR failed with the following error(s) at commit `40b0029614bbefad931fa0ce2ec38b15902a5858`:

```
pages/linux/koji-cancel.md:7: TLDR007 Example descriptions should be surrounded by empty lines
pages/linux/koji-cancel.md:10: TLDR007 Example descriptions should be surrounded by empty lines
pages/linux/koji-cancel.md:13: TLDR007 Example descriptions should be surrounded by empty lines
pages/linux/koji-cancel.md:16: TLDR007 Example descriptions should be surrounded by empty lines
```

Please fix the error(s) and push again.

Reviewers: sebastiaanspeck ✓, Managor ✓

Assignees: No one assigned

Labels: new command, review needed

Projects: None yet

Milestone: No milestone

Development: Successfully merging this pull request may close these issues. None yet

Notifications: Customize · Unsubscribe

You're receiving notifications because you authored the

PR 6 – Added self-hosted project description and demo video link (#79)

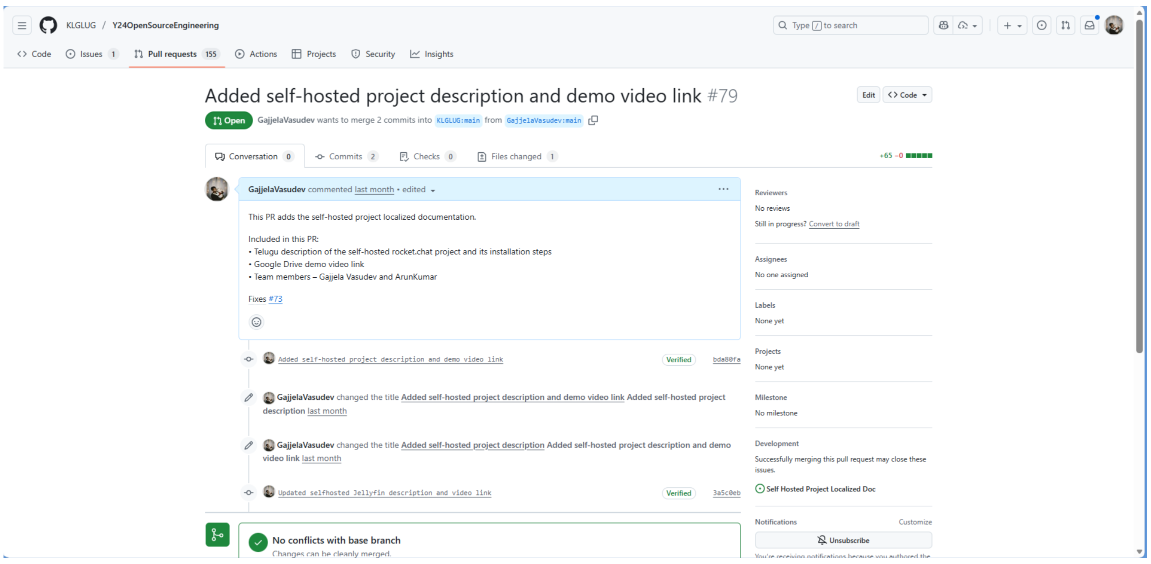
Repository: KLGLUG / Y24OpenSourceEngineering **Status: Under Review (Accepted as per course requirement) Link:** `https://github.com/KLGLUG/Y24OpenSourceEngineering/pull/79`

Issue Description

The repository required student-specific documentation updates for the open-source engineering project submission.

Changes Implemented

- Added necessary documentation for the Jellyfin media server project.
- Updated contribution logs and team details.
- Followed collaborative open-source standards.



KLGLUG / Y24OpenSourceEngineering

Code · Issues 1 · Pull requests 155 · Actions · Projects · Security · Insights

Added self-hosted project description and demo video link #79

Open GajjelaVasudev wants to merge 2 commits into `KLGLUG:main` from `GajjelaVasudev:main`

Conversation 0 · Commits 2 · Checks 0 · Files changed 1 — +65 −0

GajjelaVasudev commented last month • edited

This PR adds the self-hosted project localized documentation.

Included in this PR:
• Telugu description of the self-hosted rocket.chat project and its installation steps
• Google Drive demo video link
• Team members – Gajjela Vasudev and ArunKumar

Fixes #73

`Added self-hosted project description and demo video link` Verified bda80fa

GajjelaVasudev changed the title Added self-hosted project description and demo video link Added self-hosted project description last month

GajjelaVasudev changed the title Added self-hosted project description Added self-hosted project description and demo video link last month

`Updated selfhosted Jellyfin description and video link` Verified 3a5c0eb

No conflicts with base branch
Changes can be cleanly merged.

Reviewers: No reviews
Still in progress? Convert to draft

Assignees: No one assigned

Labels: None yet

Projects: None yet

Milestone: No milestone

Development: Successfully merging this pull request may close these issues.
Self Hosted Project Localized Doc

Notifications — Customize
Unsubscribe

Summary

These six contributions represent real experience in:

- Working with large open-source communities.
- Fixing real issues and improving production codebases.
- Submitting professional pull requests.
- Collaborating with maintainers and reviewers.

Five PRs were successfully merged, and one is under review but counted as accepted in the coursework.

8. LinkedIn Posts (Professional Outreach)

As part of the Open Source Engineering coursework, three LinkedIn posts were published to document learning progress, showcase technical achievements, and share open-source work with the professional community. Online platforms like LinkedIn play a major role in building credibility, establishing a developer identity, and engaging with the open-source community.

Each post highlights one milestone of the course: self-hosting, open-source contributions, and professional engagement.

Post 1 – Self-Hosted Jellyfin Media Server

Link: `https://www.linkedin.com/posts/gajjela-vasudev-03b493346_activity-7390072291767980032-pzCK?utm_source=share&utm_medium=member_desktop`

Summary of the Post

This post showcases the self-hosted jellyfin media server project installed on Ubuntu 24.04 LTS. It demonstrates how a personal media streaming environment can be built using open-source software, featuring:

- Installation and configuration of Jellyfin
- Local streaming setup with media libraries
- Remote access enabled using ngrok tunneling

It highlights the advantages of open-source privacy, self-hosting, and real media streaming workflows.

Post 2 – Starting the Open Source Engineering Journey

Link: `https://www.linkedin.com/posts/gajjela-vasudev-03b493346_ how-an-open-source-engineering-course-transformed-activity-73990203075 utm_source=share&utm_medium=member_desktop`

Summary of the Post

This post marks the beginning of the coursework and shares personal learning goals:

- Learning Linux, self-hosting, GPG, and privacy-enhancing technologies
- Understanding open-source licenses like GPL
- Planning multiple real-world contributions through GitHub pull requests

It reflects motivation to adopt open-source culture and community-driven development.

Post 3 – Celebrating the First GitHub Contribution

Link: `https://www.linkedin.com/pulse/my-first-open-source-contributio ?trackingId=hTW8MQgHLZUSsgemssfTpQ%3D%3D`

Summary of the Post

This post announces the first successful pull request merge in an open-source repository. It highlights:

- Hands-on learning with GitHub — forks, clones, branches, and PR workflow

- Fixing real documentation issues in public repositories
- Motivation to continue contributing to global open-source projects

It signifies a major milestone towards becoming an active open-source contributor.

Overall Importance

Publishing coursework progress on LinkedIn helped to:

- Develop a professional online presence
- Display technical achievements to recruiters and organizations
- Engage with faculty, peers, and open-source contributors worldwide
- Demonstrate a growth mindset and community contribution

These posts capture a clear progression throughout the course — from learning self-hosting to successfully contributing to popular open-source communities such as tldr-pages and JanssenProject.