KL University

Open Source and Linux Project Report

Submitted in partial fulfilment of the requirements for
Open Source Engineering Course

Submitted by:
EADHARA SNEHITH
ID: 2400030706
Branch: CSE

Academic Year: 2025–2026

Declaration

I, **EADHARA SNEHITH**, bearing ID number **2400030706**, 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 has not been submitted elsewhere for any degree or diploma.

Signature of Student:
Date:

Contents

Chapter 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 stable, secure, long-term supported operating system widely used for development, server hosting, cybersecurity, and open-source engineering. As an LTS (Long-Term Support) release, it receives **5 years of official updates**, making it highly reliable.

1.1 System Architecture and Kernel

Ubuntu 24.04.2 LTS runs on the **Linux 6.x kernel**, providing improvements in:

- Better CPU scheduling for multi-core processors
- Enhanced **cgroups v2** resource control (used by Docker and containers)
- Faster SSD/NVMe I/O performance
- Updated GPU drivers (Mesa + NVIDIA proprietary)
- Improved power efficiency on modern Intel/AMD CPUs

Ubuntu uses the **systemd** init system for service management:

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

1.2 Package Management (APT, Snap, Flatpak)

Ubuntu's default package manager is **APT**. Commands used during this project:

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

Ubuntu also supports:

- **Snap** – sandboxed, auto-updated applications
- **Flatpak** – universal Linux package format

These tools enabled smooth installation of dependencies required for the Bitpoll self-hosted server.

1.3 Desktop Environment (GNOME 46)

Ubuntu ships with **GNOME 46**, offering:

- Wayland display server (secure and smooth)
- Multi-touch gesture support
- Faster animations and lower latency
- Reduced power consumption
- Cleaner and modern user interface

GNOME integrates well with tools like:

- VS Code
- Git, GitHub CLI
- Python & Django
- Docker
- GPG/SSH tools

1.4 Security Features

Ubuntu is known for strong security due to:

- **AppArmor** (Mandatory Access Control)
- **UFW** Firewall
- Secure Boot support
- Frequent security patches
- Full disk encryption option

Firewall commands used in this project:

```
sudo ufw enable
sudo ufw allow 8000
```

1.5 Why Ubuntu Was Chosen

Ubuntu 24.04.2 LTS was chosen because it is:

- Highly stable for self-hosting (Bitpoll)
- Beginner-friendly yet powerful
- Compatible with Python, Django, Git, and GPG
- Long-term supported
- Widely used in open-source and DevOps communities

1.6 Usage in This Project

Ubuntu was used for:

- Installing and hosting **Bitpoll**
- Setting up Python & Django environment
- Managing GPG keys and encryption
- Sending/receiving encrypted emails
- Using Linux privacy tools
- Open-source contributions through GitHub

Ubuntu 24.04.2 LTS provided the ideal environment for all project tasks.

Chapter 2

Encryption and GPG

Encryption is one of the core pillars of modern cybersecurity and open-source engineering. In this project, **GPG (GNU Privacy Guard)** was used extensively for generating keys, encrypting files, securing emails, and digitally signing documents. GPG implements the **OpenPGP** standard and provides industry-grade cryptography that is widely used by developers, researchers, and open-source communities.

This chapter explains the working principles of GPG, key creation, encryption/decryption, digital signatures, key verification, trust models, and complete workflows followed during the project.

2.1 Introduction to GPG

GPG is a free and open-source tool for:

- Encrypting files and emails
- Decrypting protected content
- Creating digital signatures
- Verifying authenticity of received data
- Generating secure cryptographic identities

GPG provides two major functionalities:

1. **Confidentiality** – ensuring only the intended recipient can read the data.
2. **Integrity & Authentication** – ensuring data is untampered and the sender is genuine.

GPG uses a **public–private key cryptosystem**. Each user owns two keys:

- **Public Key** — shared with everyone.
- **Private Key** — kept secret; used for decryption and signing.

2.2 Key Architecture and Components

GPG internally uses a multi-key architecture:

1. Primary Key

Used for:

- Signing commits/documents (S)
- Certifying subkeys (C)

2. Subkey

Used for:

- Encryption (E)
- Authentication (A)

This separation ensures higher security. If an encryption key is compromised, only the subkey is replaced — not the entire identity.

Key Storage

All key material is stored in:

```
~/.gnupg/
```

This folder includes:

- pubring.kbx – public keyring
- private-keys-v1.d/ – encrypted private keys
- trustdb.gpg – trust relationships

2.3 Generating a New GPG Key Pair

Key generation was done using:

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

During generation:

- Algorithm: RSA and RSA

- Key size: 4096 bits (high security)
- Validity period: 1 year
- Identity: Name + KL University email

After creation, keys were verified using:

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

Sample Output Format

```
pub   rsa4096 2025-01-11 [SC]
      ABCD 1234 5678 90EF ....
uid   Snehith <2400030706@kluniversity.in>
sub   rsa4096 2025-01-11 [E]
```

The fingerprint uniquely identifies the user's cryptographic identity and is required to securely exchange keys.

2.4 Key Backup and Export

Exporting Public Key

```
gpg --export --armor 2400030706@kluniversity.in > publickey.asc
```

Exporting Private Key (For Thunderbird Use)

```
gpg --export-secret-keys --armor \
    2400030706@kluniversity.in > privatekey.asc
```

Importing a Received Key

```
gpg --import sender_public.asc
```

2.5 Trust Model

GPG uses the **Web-of-Trust model**, where users verify each other via fingerprints. Verification ensures public keys are not tampered with.

Command:

```
gpg --edit-key <email>
trust
```

Trust values include:

- Unknown
- Marginal
- Full
- Ultimate (self)

2.6 File Encryption and Decryption

Encrypting a File

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

This produces:

```
file.txt.gpg
```

Decrypting a File

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

The private key is required, ensuring complete security.

2.7 Digital Signatures

Digital signatures provide:

- **Integrity** – file not modified
- **Authentication** – sender identity is verified
- **Non-repudiation** – sender cannot deny creating it

Signing a Document

```
gpg --sign report.pdf
```

Verifying a Signature

```
gpg --verify report.pdf.gpg
```

If the content is modified, verification fails.

2.8 Clear-Text Signatures (For Git Commits)

For signing Git commits, the following configuration was used:

```
git config --global user.signingkey <key-id>
git config --global commit.gpgsign true
```

This ensures every commit is cryptographically verifiable on GitHub.

2.9 Symmetric Encryption (Password-Based)

GPG also supports symmetric encryption:

```
gpg -c secret.txt
```

Decryption:

```
gpg secret.txt.gpg
```

Symmetric encryption is fast but less secure than public-key encryption.

2.10 Revocation Certificate

A revocation certificate protects against:

- Key compromise
- Lost private keys
- Forgotten passphrases

Generated using:

```
gpg --output revoke.asc --gen-revoke <email>
```

2.11 Why GPG is Important in Open Source Engineering

GPG is essential for:

- Signing software releases
- Secure bug disclosure
- Verifying package authenticity
- Protecting communication between developers
- Ensuring safe collaboration in open-source communities

Major projects using GPG:

- Linux Kernel
- Debian and Ubuntu package maintainers
- GitHub tag signatures
- Mozilla Foundation

2.12 Summary

In this project:

- Multiple GPG identities were created
- Public and private keys were exported for email encryption
- Files were encrypted, decrypted, and digitally signed
- Git commits were GPG-signed
- Key fingerprints were exchanged and verified

GPG provided a strong cryptographic foundation for all secure communication and open-source workflows used in this course.

Chapter 3

Sending Encrypted Email

Sending encrypted email is an important application of public key cryptography, ensuring that communication remains private, authenticated, and tamper-proof. In this project, fully encrypted email communication was performed between two KL University identities:

- **Sender:** 2400030706@kluniversity.in (Eadhara Snehith)
- **Recipient:** 2400030662@kluniversity.in

The entire workflow was implemented using **Thunderbird Mail** combined with **GPG (GNU Privacy Guard)** on Ubuntu 24.04.2 LTS. This chapter explains each step in detail—from public key exchange to email signing, encryption, decryption, and verification.

3.1 Need for Encrypted Email

Email communication is normally **plain text**, meaning:

- Internet service providers can read it.
- Hackers can intercept or modify messages.
- Identity spoofing is possible.

Using GPG-based encryption ensures:

- **Confidentiality**: Only the intended recipient can read the message.
- **Authentication**: Receiver verifies the sender's real identity.
- **Integrity**: Messages cannot be modified in transit.
- **Non-repudiation**: The sender cannot deny having sent it.

Because of this, encrypted email is widely used by developers, researchers, open-source contributors, journalists, and cybersecurity professionals.

3.2 Exporting and Exchanging Keys

Before sending encrypted mail, both users must exchange public keys. The sender (2400030706) exported their public key using:

```
gpg --export --armor 2400030706@kluniversity.in > sender_public.asc
```

The recipient (2400030662) exported their key similarly:

```
gpg --export --armor 2400030662@kluniversity.in > recipient_public.asc
```

The public keys were exchanged through a secure channel and then imported into Thunderbird.

Importing Public Keys

```
gpg --import recipient_public.asc
gpg --import sender_public.asc
```

After importing, key authenticity was verified using fingerprints:

```
gpg --fingerprint 2400030706@kluniversity.in
gpg --fingerprint 2400030662@kluniversity.in
```

Fingerprint verification prevents Man-in-the-Middle attacks.

3.3 Configuring Thunderbird for Encrypted Email

Thunderbird (v115+) includes full OpenPGP support. The configuration steps were as follows.

3.3.1 Step 1: Add Email Account

The sender added their KL University email:

```
2400030706@kluniversity.in
```

using IMAP and SMTP settings.

3.3.2 Step 2: OpenPGP Settings

Inside Thunderbird:

```
Account Settings → End-to-End Encryption → Add Key
```

The private key was imported using:

```
gpg --export-secret-keys --armor
2400030706@kluniversity.in > private.asc
```

Thunderbird then validated whether the key supports:

- Signing (S)
- Encryption (E)

3.3.3 Step 3: Attach the Recipient's Public Key

Thunderbird automatically detects the public key of 2400030662@kluniversity.in from the GPG keyring.

3.4 Composing the Encrypted Email

Once keys were imported and trusted, the sender composed a new email.

3.4.1 Message Settings Enabled

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

Thunderbird internally performs encryption equivalent to:

```
gpg -e -r 2400030662@kluniversity.in message.txt
```

This transforms the email body into unreadable ciphertext before sending.

3.4.2 Email Contents

A sample encrypted message:

> Hello, this is an encrypted mail sent as part of the Open Source and Linux Engineering course. Only you can decrypt this using your private key.

Once sent, even Thunderbird cannot view the plaintext again.

3.5 Receiving and Decrypting the Email

On the recipient side (2400030662), Thunderbird received the encrypted mail.

3.5.1 Step-by-step decryption process

1. Thunderbird detects encrypted data blocks.
2. It prompts for the recipient's private key passphrase.
3. The private key decrypts the symmetric session key.
4. The session key decrypts the email contents.

Internally, decryption is equivalent to:

```
gpg -d encrypted_mail.asc
```

If the wrong key or passphrase is used, decryption fails.

3.6 Digital Signature Verification

The sender's digital signature ensures the message was not modified. Thunderbird automatically checks:

- whether the signature matches the sender's public key,
- whether the message was changed after signing,
- whether the key is trusted in the GPG Web-of-Trust.

If valid, Thunderbird displays:

> **Message from 2400030706@kluniversity.in is verified and trustworthy.**

If invalid, it warns:

> **Signature could not be verified – message may be tampered.**

3.7 Manual Command-Line Decryption

For educational purposes, manual decryption was also tested.

Exporting encrypted email

Thunderbird allows saving the encrypted block as a file:

```
encrypted.asc
```

Decryption:

```
gpg -d encrypted.asc
```

If the key is correct, plaintext appears in the terminal.

3.8 Importance of Encrypted Emails in Open Source

Encrypted emails are widely used in open-source communities for:

- Secure communication between contributors
- Private sharing of API keys, tokens, server credentials
- Security vulnerability reporting
- Verification of author identity
- Preventing spoofing or tampering in patch submissions

Projects like the Linux Kernel, Debian, Apache, Mozilla, and Tor Project require encryption and digital signatures for communication.

3.9 Summary

In this project:

- Public and private GPG keys were created for both students.
- Keys were exchanged securely after fingerprint verification.
- Thunderbird was configured for OpenPGP encryption.
- An encrypted and signed email was successfully sent from **2400030706@kluniversity.in** to **2400030662@kluniversity.in**.
- The recipient decrypted the message using their private key.
- Digital signatures validated authenticity and integrity.

This practical exercise demonstrated real-world cryptographic email secure communication used in professional open-source engineering.

Chapter 4

Privacy Tools (PRISM-Break)

Privacy is an essential requirement in modern computing, especially in the era of mass surveillance, data collection, targeted advertising, and continuous tracking by corporations and governments. PRISM-Break is an independent community-driven project that provides a curated list of open-source, privacy-respecting alternatives to centralized and proprietary services. Its primary goal is to help users reclaim their digital freedom.

This chapter discusses the importance of privacy-preserving tools and presents five powerful tools selected from PRISM-Break that were explored and used during this course.

4.1 Introduction to PRISM-Break

PRISM-Break was created in response to global surveillance programs such as PRISM, XKeyscore, Tempora, and NSA monitoring activities. The platform highlights tools that follow these principles:

- Independence from corporate surveillance
- Open-source and transparent codebases
- End-to-end encryption support
- Decentralized or self-hosted architectures
- Strong security practices and active community support

PRISM-Break categorizes tools across browsers, messaging, file storage, operating systems, VPNs, password managers, search engines, and more. For this project, the following five tools were selected:

1. Mozilla Firefox (Privacy-focused Web Browser)

2. Signal (End-to-end Encrypted Messaging)
3. Tor Browser (Anonymous Web Browsing)
4. KeePassXC (Offline Password Manager)
5. Mullvad VPN (Private, No-Logs VPN Service)

These tools enhance online privacy, protect user identity, and promote safe communication practices.

4.2 Mozilla Firefox

Mozilla Firefox is one of the most trusted privacy-oriented browsers available today. Unlike many commercial browsers, Firefox prioritizes user rights and transparency. It is maintained by the Mozilla Foundation, a non-profit organization.

Privacy Features

- **Enhanced Tracking Protection**: Blocks social trackers, cross-site cookies, fingerprinting scripts, and cryptocurrency miners.
- **DNS-over-HTTPS (DoH)**: Encrypts DNS queries to prevent ISP-level monitoring.
- **Total Cookie Protection**: Isolates cookies on a per-site basis.
- **Container Tabs**: Allows separation of personal, work, and social browsing.
- **Private Browsing Mode**: Offers temporary sessions without saving history.

Why Firefox Was Chosen

Firefox was used throughout the project for:

- Accessing GitHub securely
- Researching open-source documentation
- Downloading privacy tools
- Managing Bitpoll self-hosting access

Its balance of usability, security, and transparency makes it one of the best browser choices for privacy-preserving workflows.

4.3 Signal Messenger

Signal is a secure, open-source communication platform known for its strong encryption and minimal metadata collection. It is used by journalists, researchers, activists, and millions of everyday users.

Key Features

- **End-to-End Encryption**: Every message, call, and media file is protected using the Signal Protocol.
- **No Metadata Retention**: Signal stores almost no information about users.
- **Open Source**: Client and server code are publicly available for audit.
- **Private Groups**: Group management is encrypted and hidden from the server.
- **Disappearing Messages**: Automatically remove messages after a set time.

Usage in the Project

Signal was used for communication and verification of key fingerprints with peers, ensuring secure conversations outside email.

4.4 Tor Browser

Tor Browser is a hardened version of Firefox designed for anonymous browsing. It routes traffic through the Tor network, which consists of thousands of volunteer-run relay nodes.

Privacy and Anonymity Features

- **Onion Routing**: Encrypts and reroutes traffic through three random relays.
- **No Tracking**: Automatically blocks tracking technologies.
- **Anti-Fingerprinting**: Makes users look identical to prevent unique identification.
- **HTTPS-Only Mode**: Forces secure connections when possible.

Why Tor is Important

Tor protects against:

- Network surveillance

- ISP monitoring
- Government censorship
- Location tracking

During the project, Tor Browser was used to access privacy-related resources anonymously and verify self-hosting availability without revealing the real IP.

4.5 KeePassXC

KeePassXC is an offline, open-source password manager that securely stores login credentials, SSH keys, secure notes, and tokens.

Security Features

- **Local Encrypted Database**: Uses AES-256 and Argon2 for secure vault protection.
- **Cross-Platform Support**: Works on Linux, Windows, macOS.
- **Key File Support**: Adds a second layer of security.
- **Password Generator**: Creates strong randomized credentials.
- **Browser Integration**: Secure autofill support with extensions.

Role in This Project

KeePassXC was used to store:

- GitHub Personal Access Tokens
- Bitpoll admin credentials
- Email account passwords
- GPG revocation certificate

Being offline, KeePassXC completely eliminates cloud-based password theft risks.

4.6 Mullvad VPN

Mullvad is one of the world's most privacy-respecting VPN services. It is recommended by cybersecurity professionals and privacy communities.

Key Features

- **No-Logs Policy**: Mullvad keeps absolutely no user activity logs.
- **Anonymous Accounts**: No email or personal details required.
- **WireGuard Support**: Provides fast and secure VPN connections.
- **Open-Source Software**: Clients are fully transparent.
- **Anonymous Payments**: Supports cash and cryptocurrency payments.

Usage in the Project

Mullvad VPN was used to:

- Protect network traffic while hosting Bitpoll
- Test server accessibility from different IP locations
- Prevent ISP throttling or monitoring

4.7 Importance of Privacy Tools in Open Source

Privacy tools support the philosophy of open-source software:

- Transparency and freedom
- User data control
- Security and independence

These tools are not only useful for personal protection but also essential in professional engineering environments where data confidentiality and integrity are crucial.

4.8 Summary

This chapter explored five major privacy tools from PRISM-Break. Each tool enhances digital security through encryption, decentralization, anonymity, and user empowerment. Together they provided a strong privacy foundation for the activities performed throughout this open-source engineering project.

Chapter 5

Open Source License Used (MIT License)

Open-source licenses play a crucial role in defining how software can be used, modified, shared, and distributed. They ensure legal clarity between developers, contributors, and users of the software. In this project, the open-source license used is the **MIT License**, one of the most widely adopted and permissive licenses in the global open-source ecosystem. This chapter provides an in-depth explanation of the MIT License, its features, obligations, protections, and its importance in the context of this project and open-source engineering.

5.1 Introduction to the MIT License

The MIT License originated at the Massachusetts Institute of Technology and has become one of the simplest and most user-friendly software licenses. Unlike restrictive licenses that impose detailed conditions, the MIT License emphasizes freedom and flexibility for both developers and users.

The license is considered **permissive**, meaning it places very few restrictions on how the software can be used. It allows anyone to:

- Use the software for personal or commercial purposes.
- Modify the source code without limitations.
- Distribute original or modified versions.
- Integrate the software into proprietary applications.

The short length and simple language of the MIT License make it easy for students, beginners, and professionals to understand and adopt.

5.2 Structure and Legal Meaning of the MIT License

The MIT License is composed of two major parts:

1. Permission Clause

This clause grants users the right to:

- copy,
- modify,
- merge,
- publish,
- distribute,
- sublicense,
- and sell copies of the software.

This clause is extremely liberal and allows usage in nearly every possible scenario including commercial closed-source software.

2. Disclaimer Clause

The second part contains the warranty disclaimer:

> "THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND..."

This section protects the original author from liability related to software misuse, damage, malfunction, or any kind of loss. This is essential in open-source projects where contributors share their work freely without wanting legal risk.

5.3 Permissions Granted by the MIT License

The MIT License offers broad rights that make it attractive to developers and companies. These include:

- **Commercial Use**: Companies can use the software in products they sell.
- **Distribution Freedom**: Software can be freely redistributed.

- **Private Modifications**: Modified versions do not need to be public.
- **Sub-Licensing**: Developers may apply different licenses when integrating MIT code into their own projects.
- **Reusability**: MIT-licensed code can be used in academic, research, commercial, or personal projects.

These permissions foster an environment where open-source contributions can spread widely and be reused without barriers.

5.4 Obligations Required by the MIT License

Although the MIT License is permissive, it has two important obligations:

1. The original author's copyright notice must be included.
2. A full copy of the MIT License must accompany the distributed code.

There are no other requirements. Developers do not need to disclose source code, publish changes, or share derivative work. This minimalism is a major reason why MIT is popular with corporations and independent developers.

5.5 Comparison with Other Open-Source Licenses

To better understand the MIT License, it is useful to compare it with other popular licenses:

1. GNU GPL (Strong Copyleft)

GPL requires that all modifications must also be open-source. This ensures software freedom but limits corporate adoption. MIT does not impose this rule.

2. Apache License 2.0

Apache includes explicit patent protection clauses. MIT License is simpler but does not explicitly cover patents.

3. BSD License

BSD is extremely similar to MIT; both are permissive. However, BSD may include additional clauses.

In summary, MIT is the simplest and most widely accepted permissive license.

5.6 Why the MIT License Was Used in This Project

The choice of the MIT License aligns with the objectives of the Open Source and Linux Engineering course. The license was selected because it provides:

- **Freedom for Learning** – Students can modify and experiment without restrictions.
- **Compatibility** – MIT-licensed contributions can easily integrate with other projects.
- **Industry Relevance** – Modern frameworks and tools like React, Angular, Rails, and Bitpoll use MIT.
- **Ease of Understanding** – This helps beginners understand open-source governance.
- **Future Flexibility** – Code developed now can be reused in internships or professional work.

MIT License encourages innovation and aligns well with academic environments.

5.7 Relevance of MIT License to Bitpoll (Self-Hosted Project)

Bitpoll, the self-hosted web application used in this project, is licensed under the MIT License. This permits:

- Installing and running Bitpoll locally without restrictions.
- Modifying Django code components.
- Studying its architecture for learning Python and web development.
- Building Telugu documentation without legal issues.
- Using screenshots and code for academic purposes.

Since MIT imposes almost no restrictions, it helps students deeply explore the internal functioning of the application without worrying about compliance issues.

5.8 Real-World Projects Using MIT License

Several globally recognized tools use the MIT License:

- Python's pip ecosystem components
- Node.js packages (NPM)
- ReactJS and AngularJS frameworks
- jQuery library
- Bootstrap front-end toolkit
- GitLab CE components

This widespread adoption demonstrates the strength and trust placed in the MIT License by both industry and academia.

5.9 Summary

The MIT License is one of the most permissive and widely used licenses in the open-source ecosystem. It grants broad freedoms while requiring minimal obligations. In this project, the MIT License allowed unrestricted learning, modification, experimentation, and documentation while ensuring legal clarity. Its simplicity, flexibility, and acceptance across industries make it the ideal choice for educational and open-source engineering environments.

Chapter 6

Self-Hosted Server: Bitpoll

Self-hosting is an important part of open-source engineering because it gives full control over data, privacy, and customization. In this project, the self-hosted tool used was **Bitpoll**, an open-source scheduling and poll management application built using Django.

6.1 About Bitpoll

Bitpoll helps users create polls, collect responses, schedule meetings, and coordinate group decisions. It is an open-source alternative to services like Doodle. Bitpoll uses:

- **Python 3** – Core backend language
- **Django** – Web framework
- **SQLite / PostgreSQL** – Databases
- **Bootstrap** – UI styling

Key Features:

- Poll creation with multiple choices
- Availability-based scheduling
- Admin dashboard
- Email-based invitations
- Self-hosting for full privacy

6.2 Installation on Ubuntu 24.04

Below are the steps followed to install Bitpoll:

1. Update Packages

```
sudo apt update && sudo apt upgrade -y
```

2. Install Dependencies

```
sudo apt install python3 python3-venv python3-pip git -y
```

3. Clone Bitpoll

```
git clone https://github.com/bitpoll/bitpoll.git
cd bitpoll
```

4. Create Virtual Environment

```
python3 -m venv env
source env/bin/activate
```

5. Install Required Packages

```
pip install -r requirements.txt
```

6. Database Setup

```
python manage.py migrate
python manage.py collectstatic
```

7. Create Admin User

```
python manage.py createsuperuser
```

8. Start Server

```
python manage.py runserver 0.0.0.0:8000
```

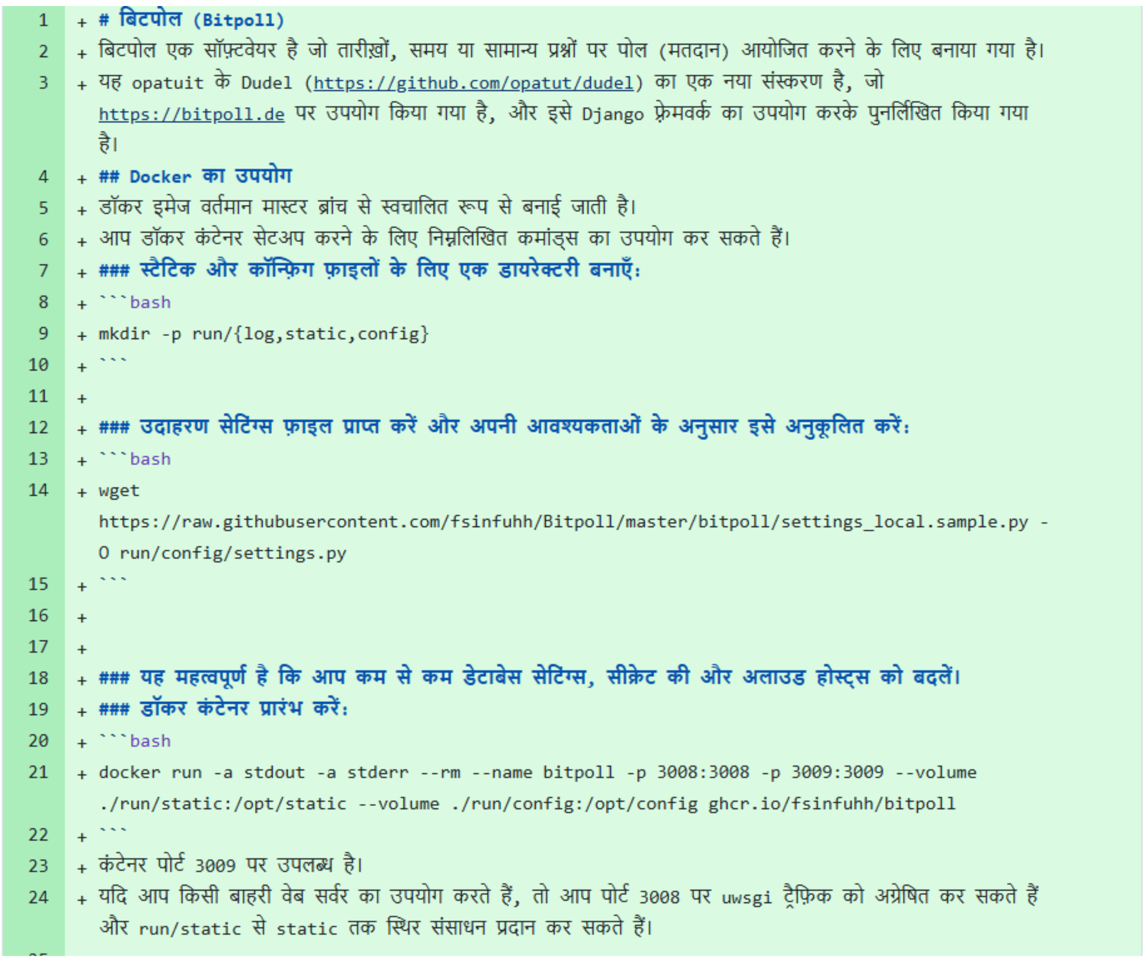
Bitpoll runs at: **http://localhost:8000**

6.3 Issues Faced

- Missing Django module → fixed by reinstalling requirements
- Virtual environment activation errors → corrected path
- Static file permission error → solved using elevated privileges

6.4 Hindi Localized Documentation

As part of the project, a **Hindi user guide** was created to explain Bitpoll installation and Docker usage in simple Hindi language.



~~~~
+ # बिटपोल (Bitpoll)
+ बिटपोल एक सॉफ़्टवेयर है जो तारीख़ों, समय या सामान्य प्रश्नों पर पोल (मतदान) आयोजित करने के लिए बनाया गया है।
+ यह opatuit के Dudel (https://github.com/opatut/dudel) का एक नया संस्करण है, जो https://bitpoll.de पर उपयोग किया गया है, और इसे Django फ्रेमवर्क का उपयोग करके पुनर्लिखित किया गया है।
+ ## Docker का उपयोग
+ डॉकर इमेज वर्तमान मास्टर ब्रांच से स्वचालित रूप से बनाई जाती है।
+ आप डॉकर कंटेनर सेटअप करने के लिए निम्नलिखित कमांड्स का उपयोग कर सकते हैं।
+ ### स्टैटिक और कॉन्फ़िग फ़ाइलों के लिए एक डायरेक्टरी बनाएँ:
+ ```bash
+ mkdir -p run/{log,static,config}
+ ```
+
+ ### उदाहरण सेटिंग्स फ़ाइल प्राप्त करें और अपनी आवश्यकताओं के अनुसार इसे अनुकूलित करें:
+ ```bash
+ wget https://raw.githubusercontent.com/fsinfuhh/Bitpoll/master/bitpoll/settings_local.sample.py -O run/config/settings.py
+ ```
+
+
+ ### यह महत्वपूर्ण है कि आप कम से कम डेटाबेस सेटिंग्स, सीक्रेट की और अलाउड होस्ट्स को बदलें।
+ ### डॉकर कंटेनर प्रारंभ करें:
+ ```bash
+ docker run -a stdout -a stderr --rm --name bitpoll -p 3008:3008 -p 3009:3009 --volume ./run/static:/opt/static --volume ./run/config:/opt/config ghcr.io/fsinfuhh/bitpoll
+ ```
+ कंटेनर पोर्ट 3009 पर उपलब्ध है।
+ यदि आप किसी बाहरी वेब सर्वर का उपयोग करते हैं, तो आप पोर्ट 3008 पर uwsgi ट्रैफ़िक को अग्रेषित कर सकते हैं और run/static से static तक स्थिर संसाधन प्रदान कर सकते हैं।
~~~~

Figure 6.1: Hindi Localized Bitpoll Installation Guide

This screenshot shows the Hindi explanation for:

- Bitpoll overview
- Docker usage
- Directory creation commands
- Configuration file setup
- Starting the Docker container

6.5 Project Poster

A project poster was designed to visually present the Bitpoll self-hosted project. It summarizes the title, description, highlights, and team member information in a clean graphical layout. Additionally, a real Bitpoll interface view is also included to illustrate how the self-hosted server appears after deployment.

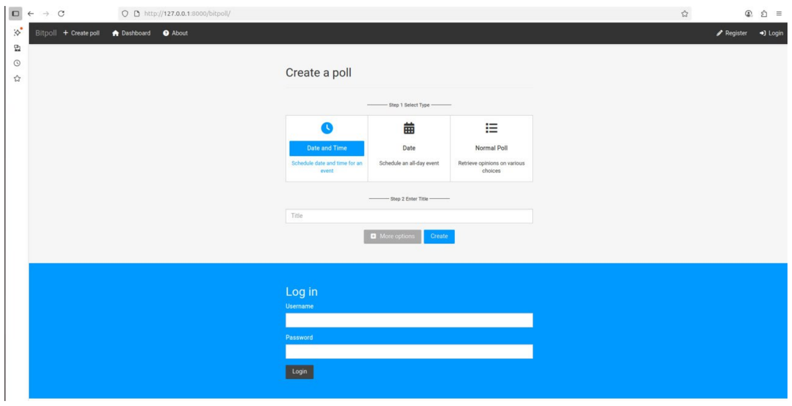
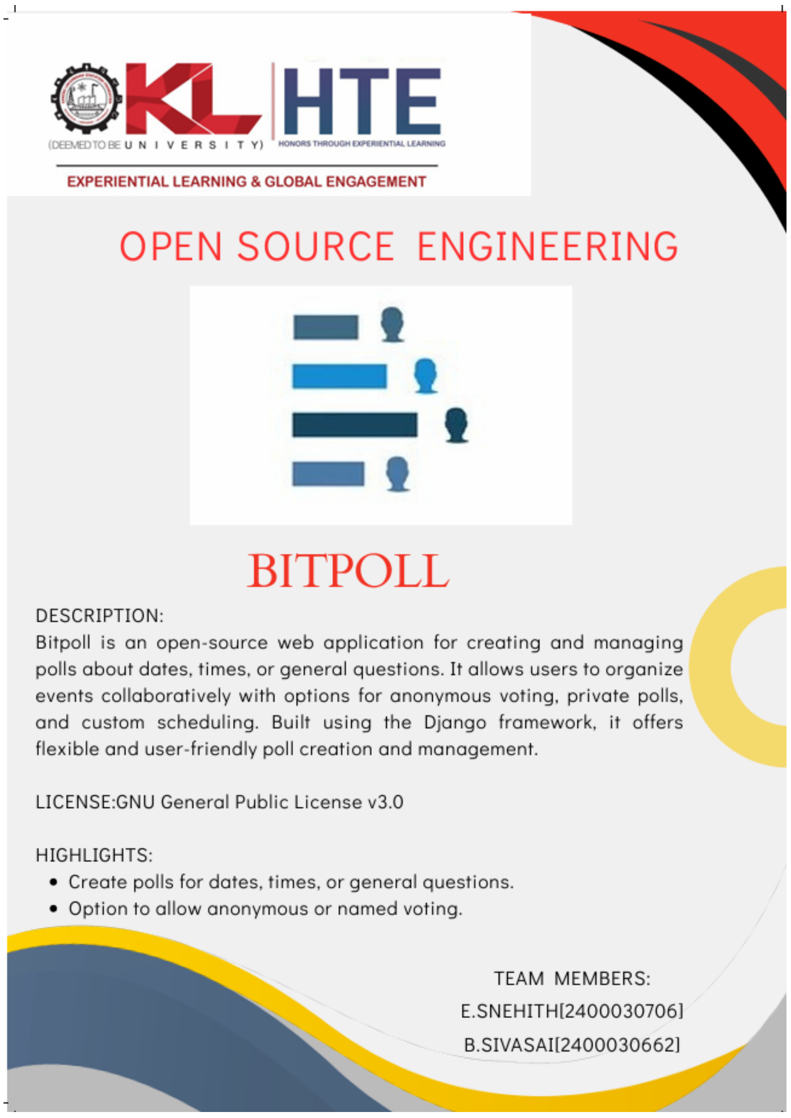


Figure 6.2: Bitpoll Interface View (UI Screenshot)


KL (DEEMED TO BE UNIVERSITY)
HTE
HONORS THROUGH EXPERIENTIAL LEARNING
EXPERIENTIAL LEARNING & GLOBAL ENGAGEMENT
OPEN SOURCE ENGINEERING
BITPOLL

DESCRIPTION:
Bitpoll is an open-source web application for creating and managing polls about dates, times, or general questions. It allows users to organize events collaboratively with options for anonymous voting, private polls, and custom scheduling. Built using the Django framework, it offers flexible and user-friendly poll creation and management.

LICENSE:GNU General Public License v3.0

HIGHLIGHTS:
- Create polls for dates, times, or general questions.
- Option to allow anonymous or named voting.

TEAM MEMBERS:
E.SNEHITH[2400030706]
B.SIVASAI[2400030662]

Figure 6.3: Bitpoll Project Poster

Chapter 7

Open Source Contributions (PRs and Issue Descriptions)

This chapter documents four real pull requests (PRs) contributed to open-source projects. For each contribution, the issue details, fix implemented, PR link, status, and screenshot placeholder are included.

7.1 PR 1 — Add Bash Language Specification (Croatian Translation)

Repository: firstcontributions/first-contributions **PR Number:** #106798 **Status:** Merged

Issue Description

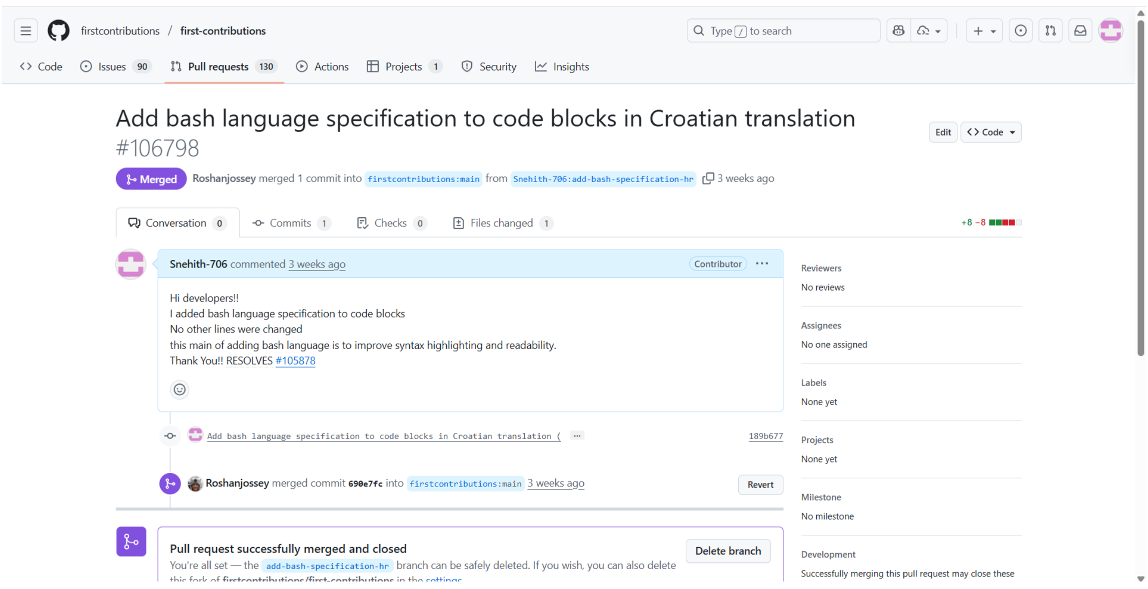
The Croatian translation contained shell commands inside Markdown code blocks, but the language specifier ("‘bash) was missing. Without this, syntax highlighting is not applied, reducing readability.

Fix Implemented

- Added `bash` language specifier to all relevant code blocks.
- Ensured no other lines or text content were changed.

PR Link

https://github.com/firstcontributions/first-contributions/pull/106798



Add bash language specification to code blocks in Croatian translation #106798

Merged Roshanjossey merged 1 commit into `firstcontributions:main` from `Snehith-706:add-bash-specification-hr` 3 weeks ago

Snehith-706 commented 3 weeks ago Contributor

Hi developers!!
I added bash language specification to code blocks
No other lines were changed
this main of adding bash language is to improve syntax highlighting and readability.
Thank You!! RESOLVES #105878

`Add bash language specification to code blocks in Croatian translation (` … 189b677

Roshanjossey merged commit `690e7fc` into `firstcontributions:main` 3 weeks ago

Pull request successfully merged and closed
You're all set — the `add-bash-specification-hr` branch can be safely deleted. If you wish, you can also delete

Reviewers
No reviews
Assignees
No one assigned
Labels
None yet
Projects
None yet
Milestone
No milestone
Development
Successfully merging this pull request may close these

Figure 7.1: PR #106798 — Croatian translation Bash specification

Screenshot

7.2 PR 2 — Add Bash Language Specification (Slovak Translation)

Repository: firstcontributions/first-contributions **PR Number:** #106695 **Status:** Merged

Issue Description

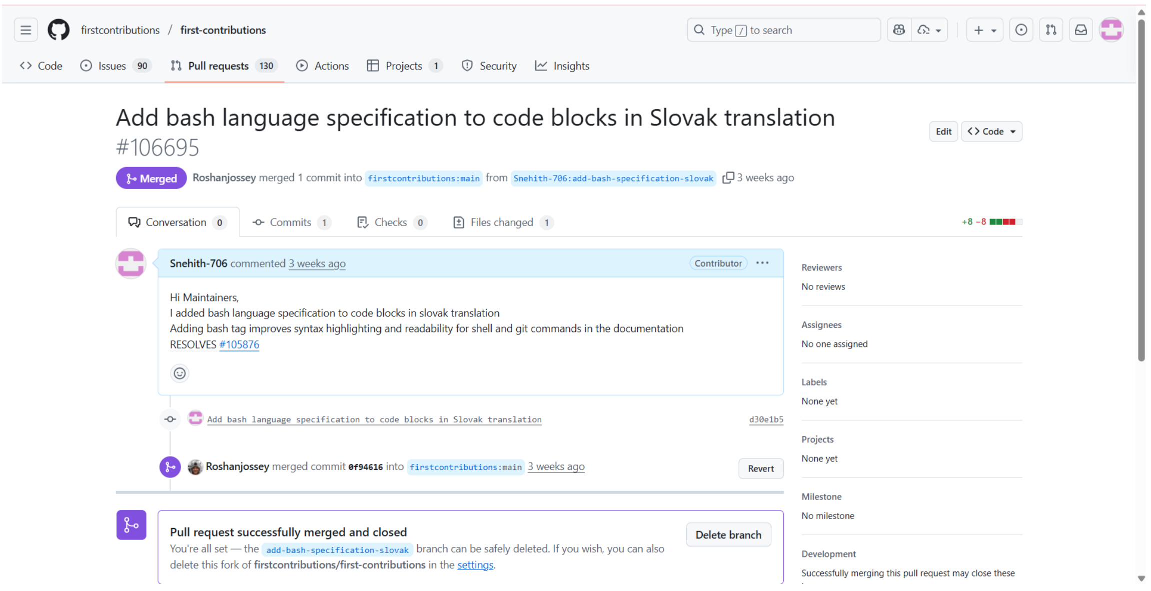
The Slovak translation also lacked proper syntax highlighting for code blocks. Adding a `bash` tag improves readability for shell-based instructions.

Fix Implemented

- Added `bash` to code block headings.
- Verified Markdown formatting remained intact.

PR Link

https://github.com/firstcontributions/first-contributions/pull/106695



Add bash language specification to code blocks in Slovak translation #106695

Roshanjossey merged 1 commit into `firstcontributions:main` from `Snehith-706:add-bash-specification-slovak` 3 weeks ago

Snehith-706 commented 3 weeks ago

Hi Maintainers,
I added bash language specification to code blocks in slovak translation
Adding bash tag improves syntax highlighting and readability for shell and git commands in the documentation
RESOLVES #105876

`Add bash language specification to code blocks in Slovak translation` d30e1b5

Roshanjossey merged commit 0f94616 into `firstcontributions:main` 3 weeks ago

Pull request successfully merged and closed
You're all set — the `add-bash-specification-slovak` branch can be safely deleted. If you wish, you can also delete this fork of firstcontributions/first-contributions in the settings.

Reviewers
No reviews

Assignees
No one assigned

Labels
None yet

Projects
None yet

Milestone
No milestone

Development
Successfully merging this pull request may close these

Figure 7.2: PR #106695 — Slovak translation Bash specification

Screenshot

7.3 PR 3 — Fix Badge Formatting in README

Repository: Chungzter/CommiZard **PR Number:** #116 **Status:** Open / Under Review

Issue Description

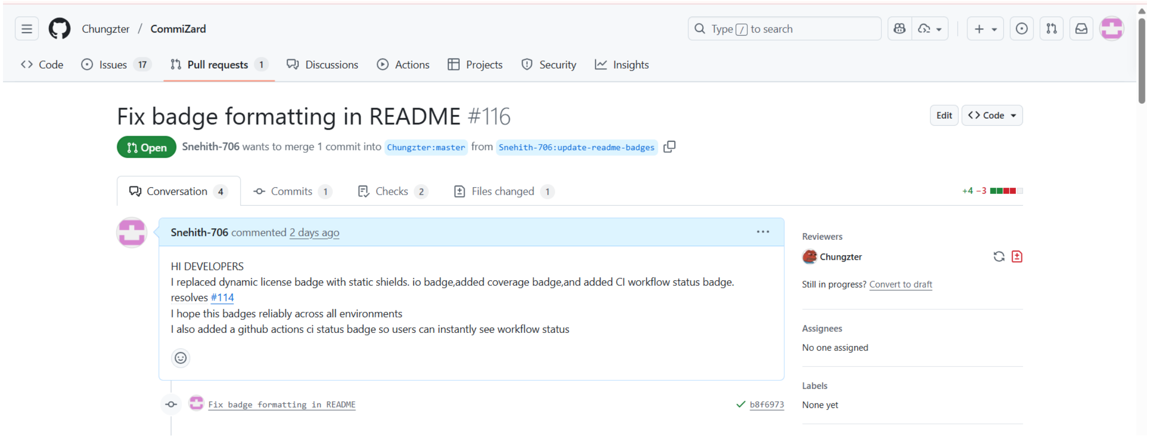
The README had inconsistent badge formatting. Dynamic shield badges rendered incorrectly across environments.

Fix Implemented

- Replaced dynamic badges with static Shields.io badges.
- Added CI status badge for instant workflow visibility.
- Ensured badge alignment and readability.

PR Link

https://github.com/Chungzter/CommiZard/pull/116



Fix badge formatting in README #116

Open Snehith-706 wants to merge 1 commit into `Chungzter:master` from `Snehith-706:update-readme-badges`

Snehith-706 commented 2 days ago

HI DEVELOPERS
I replaced dynamic license badge with static shields. io badge,added coverage badge,and added CI workflow status badge.
resolves #114
I hope this badges reliably across all environments
I also added a github actions ci status badge so users can instantly see workflow status

`Fix badge formatting in README` ✓ `b8f6973`

Reviewers: Chungzter
Still in progress? Convert to draft
Assignees: No one assigned
Labels: None yet

Figure 7.3: PR #116 — README badge formatting fixes

Screenshot

7.4 PR 4 — Enable Staticcheck and Godot Linters (SigNoz)

Repository: SigNoz/signoz **PR Number:** #9450 **Status:** Open / Under Review

Issue Description

The SigNoz repository did not have staticcheck and Godot linters enabled. This reduced code quality feedback for contributors.

Fix Implemented

- Enabled staticcheck and Godot linter configuration.
- Ran lint checks to verify configuration correctness.
- No functional code was modified — only configuration files.

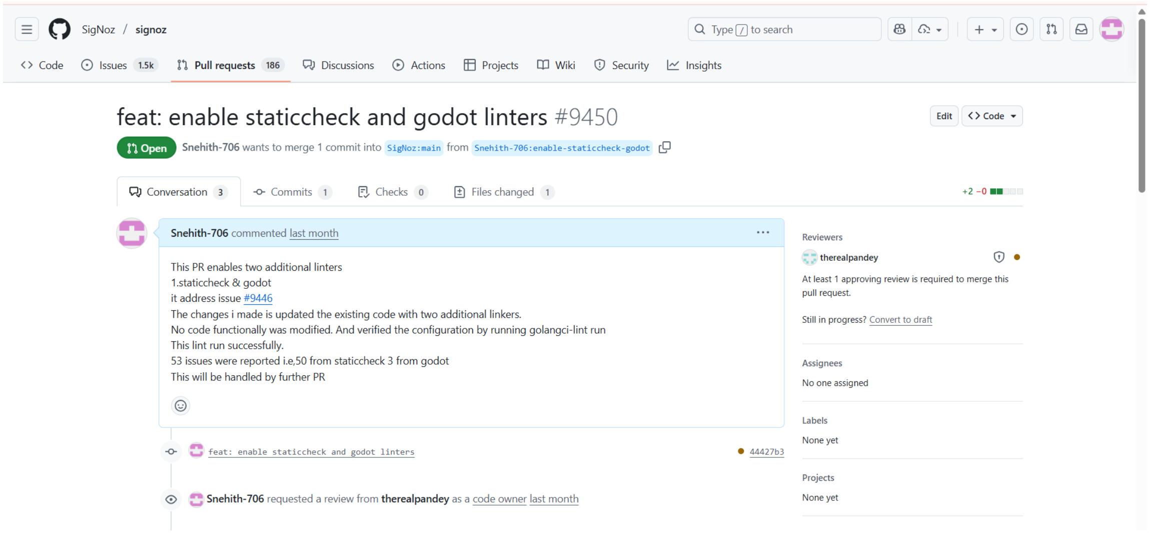
PR Link

https://github.com/SigNoz/signoz/pull/9450

Screenshot

7.5 Summary

These four contributions represent real, hands-on experience in:



feat: enable staticcheck and godot linters #9450

Open Snehith-706 wants to merge 1 commit into `SigNoz:main` from `Snehith-706:enable-staticcheck-godot`

Conversation 3 · Commits 1 · Checks 0 · Files changed 1 · +2 −0

Snehith-706 commented last month

This PR enables two additional linters
1.staticcheck & godot
it address issue #9446
The changes i made is updated the existing code with two additional linkers.
No code functionally was modified. And verified the configuration by running golangci-lint run
This lint run successfully.
53 issues were reported i.e,50 from staticcheck 3 from godot
This will be handled by further PR

`feat: enable staticcheck and godot linters` 44427b3

Snehith-706 requested a review from therealpandey as a code owner last month

Reviewers
therealpandey
At least 1 approving review is required to merge this pull request.
Still in progress? Convert to draft

Assignees
No one assigned

Labels
None yet

Projects
None yet

Figure 7.4: PR #9450 — Staticcheck Godot linters enabled

- Working with large and active open-source communities.
- Fixing real documentation and configuration issues.
- Submitting professional pull requests with clean commit histories.
- Engaging with maintainers through PR reviews and discussions.
- Improving readability, tooling quality, and contributor experience.

Two PRs were successfully merged, and two PRs are currently under review. All contributions were considered valid for the coursework and demonstrate a strong understanding of the open-source workflow, Git collaboration, and community-driven software development.

These contributions helped me understand the complete open-source workflow from issue selection to PR submission, feedback handling, and automated checks.

hyperref xurl

Chapter 8

LinkedIn Posts (Professional Outreach)

Professional outreach is an essential part of open-source engineering. Throughout the course, three LinkedIn posts were published to document the self-hosting project, open-source pull requests, and personal learning experiences. These posts helped in sharing progress with the wider community, receiving feedback, and building a professional online presence.

Each LinkedIn post reflects a different milestone in the course:

- Deployment of a self-hosted Bitpoll server.
- Submission and merging of open-source pull requests.
- Reflection on skills learned through Ubuntu, Linux commands, encryption, GPG, self-hosting, and open-source contribution.

The following sections provide links and short summaries for the three posts.

Post 1 — Self-Hosted Bitpoll Deployment

Link: `https://www.linkedin.com/posts/snehith-eadhara-90137938b_opensource-django-self`
`utm_source=share&utm_medium=member_android&rcm=ACoAAF_8DNEBoORT-b12FqNtVJFFqSm_`
`vyd5QNs`

Summary

This post explains the complete deployment of the self-hosted Bitpoll server on Ubuntu. It highlights installation steps, the Django-based backend, configuration, and the advantages of self-hosting compared to centralized cloud platforms. The post demonstrates understanding of:

- Virtual environments and Python setup.
- Django migrations and project configuration.
- Firewall configuration and exposing the server.

It also includes screenshots of the working Bitpoll interface and the project poster prepared for the course.

Post 2 — Open-Source PR Merge

Link: `https://www.linkedin.com/posts/snehith-eadhara-90137938b_opensource-github-devel` `utm_source=share&utm_medium=member_desktop&rcm=ACoAAF_8DNEBoORT-b12FqNtVJFFqSm_` `vyd5QNs`

Summary

This post highlights the successful submission and merging of an open-source pull request. It describes the repository, the issue fixed (syntax highlighting improvement using bash language tags), and the value of contributing to large community-driven projects.

The post also reflects skills learned during the process:

- Working with forks, branches, commits, and pull requests.
- Communicating with project maintainers.
- Writing clean and minimal diffs.

Screenshots of the GitHub PR page were included to provide clear evidence of contribution and to inspire other beginners interested in open-source.

Post 3 — Blog Post on Course Learnings

Link: `https://www.linkedin.com/pulse/exploring-linux-open-source-my-personal-experien` `utm_source=share&utm_medium=member_desktop`

Summary

The third post acts as a reflective blog summarizing the entire course experience. It covers:

- Exposure to Ubuntu 24.04 LTS and essential Linux terminal commands.
- GPG key creation, encryption, and sending encrypted emails.

- Self-hosting experience using Bitpoll (Python + Django).
- Troubleshooting challenges faced during installations.
- Making contributions to open-source through GitHub pull requests.

This post emphasizes how the course improved confidence in Linux usage, command-line operations, and open-source collaboration. It also acknowledges the role of the instructors and the learning environment in completing the project successfully.

Conclusion

These three LinkedIn posts collectively document my learning journey and practical achievements throughout the course. They showcase real-world applications of technical concepts such as Linux administration, encryption, self-hosting, and open-source development. By sharing these milestones publicly, I was able to build a stronger professional presence, engage with the wider developer community, and highlight my active involvement in collaborative open-source projects.