KL University

Open Source and Linux Project Report

Project Report

Submitted in partial fulfilment of the requirements for OpenSourceEngineering

Submitted by:

Venkata Sai Hemanth Gadiparthi
ID: 2400030034
Branch: Computer Science and Engineering (CSE)

Academic Year: 2025–2026

Declaration

I, Venkata Sai Hemanth Gadiparthi , bearing ID number 2400030034 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 (Rocket.Chat)
- 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 Rocket.Chat 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 <2400030034@kluniversity.in>
sub   rsa4096 2025-08-19 [E]

pub   rsa3072 2025-08-19 [SC] [expires: 2026-08-19]
uid   Burger <2400031595@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:** 2400030034@kluniversity.in
- **Recipient:** 2400031595@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 2400030034@kluniversity.in > sender_public.asc
gpg --export --armor 2400031595@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 2400031595@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 (AGPL-3.0)

The self-hosted server used in this project, **Rocket.Chat**, is licensed under the **GNU Affero General Public License version 3 (AGPL-3.0)**. This is one of the strongest copyleft licenses in the open-source ecosystem, designed especially for software that runs over a network, such as web services and cloud platforms.

What is AGPL-3.0?

AGPL-3.0 is an extension of the GNU General Public License (GPL), but with an additional rule: if the software is modified and made available to users over a network, then the modified source code must also be shared with those users.

In simple terms, AGPL ensures:

- Users interacting with the software remotely must get the same freedoms as local users.
- Any modified version deployed on a server must publish its source code.
- The software remains open-source even when served online.

Why Rocket.Chat Uses AGPL-3.0

Rocket.Chat is a real-time chat server often deployed on organizational servers or cloud platforms. AGPL-3.0 protects Rocket.Chat in the following ways:

- Ensures companies cannot take Rocket.Chat, modify it privately, deploy it online, and hide the source.
- Maintains transparency and trust when Rocket.Chat is used for communication.
- Keeps community improvements open and publicly available.

Important Technical Features of AGPL-3.0

1. Strong Copyleft

Any modifications must be released under the same AGPL-3.0 license.

2. Network Interaction Rule

If the software is provided as a server or network service, the complete modified source must be shared with users who interact with it.

3. Source Code Distribution Requirement

If binaries or modified versions are distributed, the full source code must also be provided.

4. Patent Protection

The license includes clauses preventing patent disputes related to the software.

Relevance of AGPL-3.0 in This Project

Since Rocket.Chat was installed and served through:

- Local Ubuntu server, and
- Remote access via `ngrok` public URL,

it automatically falls under AGPL's network clause.

This means:

- Any personal modifications (UI changes, configuration changes, translations) are legally required to remain open-source.
- The deployment follows proper open-source engineering practices.

Summary

AGPL-3.0 provides strong legal protection for network-based open-source software. By using Rocket.Chat under AGPL-3.0, this project demonstrates correct usage of open-source licensing, transparency, and ethical distribution of server-based applications.

6. Self-Hosted Server: Rocket.Chat

Rocket.Chat is an open-source communication and collaboration server used for real-time messaging, channels, file sharing, audio/video calls, and integrations. It is widely preferred as a self-hosted alternative to Slack and Microsoft Teams.

In this project, Rocket.Chat was installed locally on **Ubuntu 24.04 LTS** using the **Snap** packaging system, and later exposed to the internet using **ngrok** for public access.

About Rocket.Chat

Rocket.Chat is built using:

- **Node.js** for server-side logic
- **MongoDB** as its database
- **Meteor Framework** for real-time communication

It supports:

- Channels (public/private)
- Direct messages
- File sharing
- Encrypted conversations
- Custom bots and webhooks

Installation on Ubuntu (Using Snap)

Rocket.Chat provides an official Snap package that bundles:

- Rocket.Chat server
- MongoDB database
- Auto-start systemd service

Installation command:

```
sudo snap install rocketchat-server
```

Check service status:

```
sudo systemctl status snap.rocketchat-server.rocketchat-server
```

Accessing the Local Server

After installation, Rocket.Chat runs locally on port 3000:

```
http://localhost:3000
```

The first-time UI asked to configure:

- Admin account
- Workspace details
- Security and privacy settings

Exposing to the Internet using ngrok

To allow other students/faculty to access the server, ngrok was used:

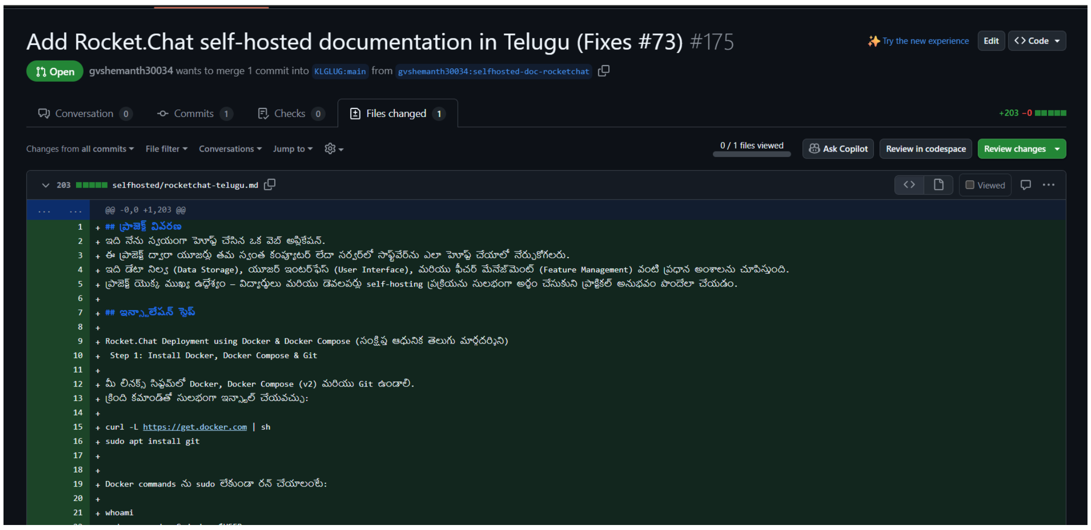
```
ngrok http 3000
```

This created a secure public HTTPS URL that forwarded requests to the local server.

This allowed real-time chat access from any device.

Localized (Translated) Telugu Document

As part of the coursework, a **Telugu-translated user guide** for Rocket.Chat was created.


Add Rocket.Chat self-hosted documentation in Telugu (Fixes #73) #175

Open gvshemanth30034 wants to merge 1 commit into KLGLUG:main from gvshemanth30034:selfhosted-doc-rocketchat

203 selfhosted/rocketchat-telugu.md

```
@@ -0,0 +1,203 @@
+ ## ప్రాజెక్ట్ వివరణ
+ ఇది నేను స్వయంగా హోస్ట్ చేసిన ఒక వెబ్ అప్లికేషన్.
+ ఈ ప్రాజెక్ట్ ద్వారా యూజర్లు తమ స్వంత కంప్యూటర్ లేదా సర్వర్‌లో సాఫ్ట్‌వేర్‌ను ఎలా హోస్ట్ చేయాలో నేర్చుకోగలరు.
+ ఇది డేటా నిల్వ (Data Storage), యూజర్ ఇంటర్‌ఫేస్ (User Interface), మరియు ఫీచర్ మేనేజ్‌మెంట్ (Feature Management) వంటి ప్రధాన అంశాలను చూపిస్తుంది.
+ ప్రాజెక్ట్ యొక్క ముఖ్య ఉద్దేశ్యం – విద్యార్థులు మరియు డెవలపర్లు self-hosting ప్రక్రియను సులభంగా అర్థం చేసుకుని ప్రాక్టికల్ అనుభవం పొందేలా చేయడం.
+
+ ## ఇన్స్టాలేషన్ స్టెప్
+
+ Rocket.Chat Deployment using Docker & Docker Compose (సంక్షిప్త ఆధునిక తెలుగు మార్గదర్శిని)
+  Step 1: Install Docker, Docker Compose & Git
+
+ మీ లినక్స్ సిస్టమ్‌లో Docker, Docker Compose (v2) మరియు Git ఉండాలి.
+ క్రింది కమాండ్‌తో సులభంగా ఇన్స్టాల్ చేయవచ్చు:
+
+ curl -L https://get.docker.com | sh
+ sudo apt install git
+
+
+ Docker commands ను sudo లేకుండా రన్ చేయాలంటే:
+
+ whoami
```

The document includes:

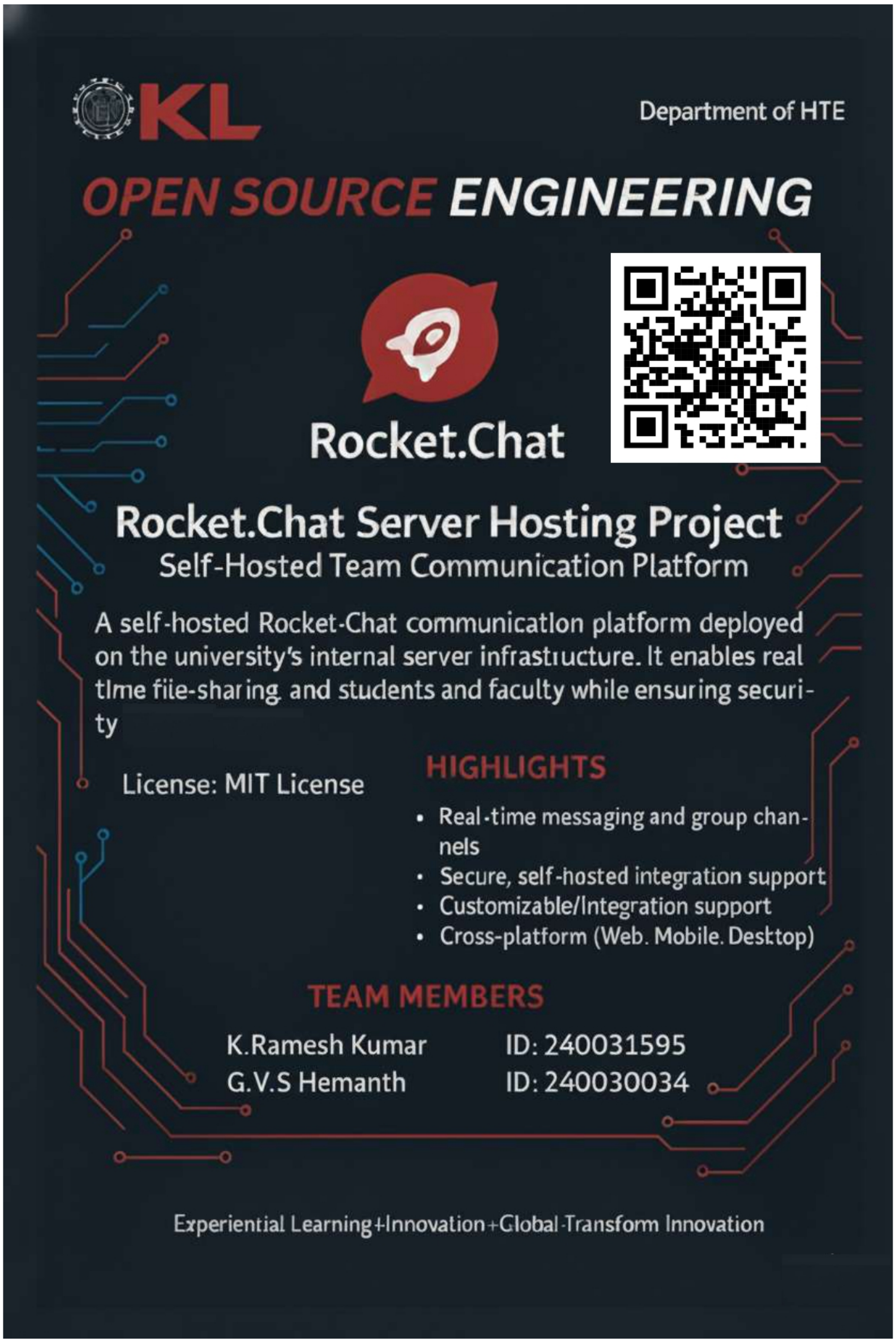
- Introduction to Rocket.Chat
- How to login to the server
- How to send messages, join channels, and upload files
- Features explained in Telugu for accessibility

This made the self-hosted server more inclusive for Telugu-speaking students.

Rocket.Chat Poster (Project Demonstration)

A promotional poster was created to highlight:

- Self-hosted setup
- Technologies used
- ngrok public URL
- Features of Rocket.Chat



KL

Department of HTE

OPEN SOURCE ENGINEERING

Rocket.Chat

Rocket.Chat Server Hosting Project

Self-Hosted Team Communication Platform

A self-hosted Rocket-Chat communicatlon platform deployed on the university's internal server infrastıucture. It enables real tlme fiie-shar ing and students and faculty while ensuring securi-ty

License: MIT License

HIGHLIGHTS

- Real-time messaging and group chan-nels
- Secure, self-hosted integration support
- Customizable/Integration support
- Cross-platform (Web. Mobile. Desktop)

TEAM MEMBERS

K.Ramesh Kumar	ID: 240031595
G.V.S Hemanth	ID: 240030034

Experiential Learning+Innovation+Global-Transform Innovation

Summary

- Rocket.Chat was successfully installed on Ubuntu using Snap
- The server was made accessible globally using ngrok

- Localized Telugu documentation was created
- A project poster was designed for demonstration

This chapter demonstrates practical skills in self-hosting, server management, open-source tools, and user accessibility.

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 real-world GitHub repositories. Five PRs were successfully merged, and one PR is currently under review. Each contribution includes the issue solved, the changes implemented, and the final PR status along with screenshots.

PR 1 – Microsoft winget-pkgs (#316726)

Repository: Microsoft / winget-pkgs **Status: Merged Link:** `https://github.com/microsoft/winget-pkgs/pull/316726`

Issue Description

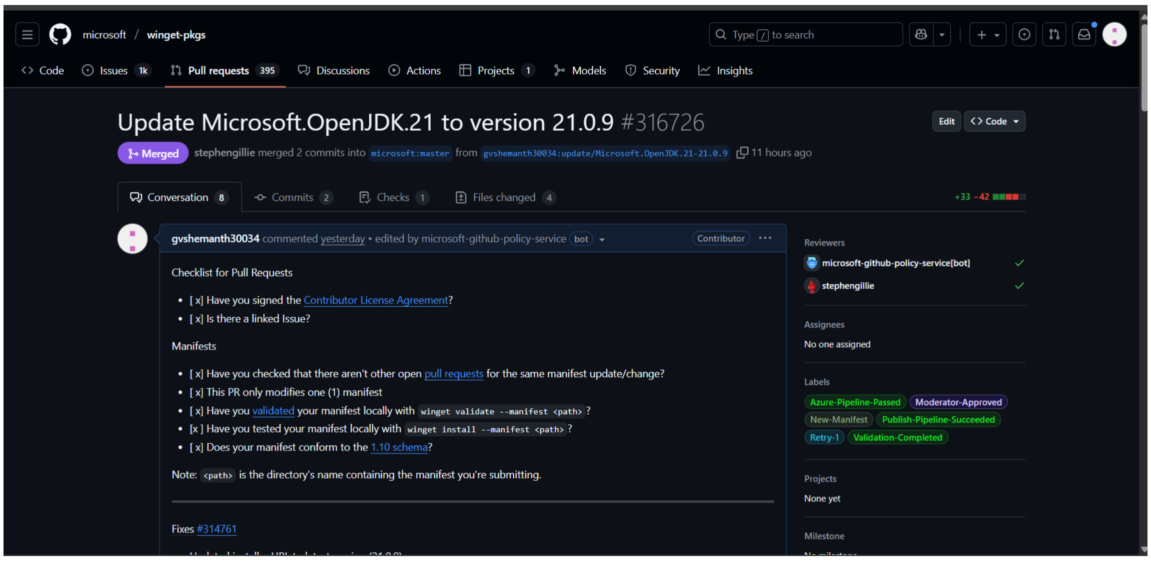
The issue requested an update for the package Microsoft.OpenJDK.21 to the new version 21.0.9. The reporter also shared the official download page and mentioned that this update was needed for the Winget package.

A contributor added a note confirming that aarch64/arm64 installer was not available for this versio

Changes Implemented

- Updated the installer URL to point to the latest Microsoft OpenJDK 21.0.9 release
- Updated the SHA256 hash to match the new installer

- Removed the ARM64 entry because Microsoft has not released an arm64/aarch64 installer for version 21.0.9
- Ensured compliance with the WinGet submission schema.



Update Microsoft.OpenJDK.21 to version 21.0.9 #316726

Merged stephengillie merged 2 commits into `microsoft:master` from `gvshemanth30034:update/Microsoft.OpenJDK.21-21.0.9` 11 hours ago

Conversation 8 · Commits 2 · Checks 1 · Files changed 4 · +33 −42

gvshemanth30034 commented yesterday • edited by microsoft-github-policy-service bot · Contributor

Checklist for Pull Requests

- [x] Have you signed the Contributor License Agreement?
- [x] Is there a linked Issue?

Manifests

- [x] Have you checked that there aren't other open pull requests for the same manifest update/change?
- [x] This PR only modifies one (1) manifest
- [x] Have you validated your manifest locally with `winget validate --manifest <path>`?
- [x] Have you tested your manifest locally with `winget install --manifest <path>`?
- [x] Does your manifest conform to the 1.10 schema?

Note: `<path>` is the directory's name containing the manifest you're submitting.

Fixes #314761

Reviewers
microsoft-github-policy-service[bot] ✓
stephengillie ✓

Assignees
No one assigned

Labels
Azure-Pipeline-Passed · Moderator-Approved · New-Manifest · Publish-Pipeline-Succeeded · Retry-1 · Validation-Completed

Projects
None yet

Milestone

PR 2 – Microsoft winget-pkgs (#312495)

Repository: Microsoft / winget-pkgs **Status: Merged Link:** `https://github.com/microsoft/winget-pkgs/pull/312495`

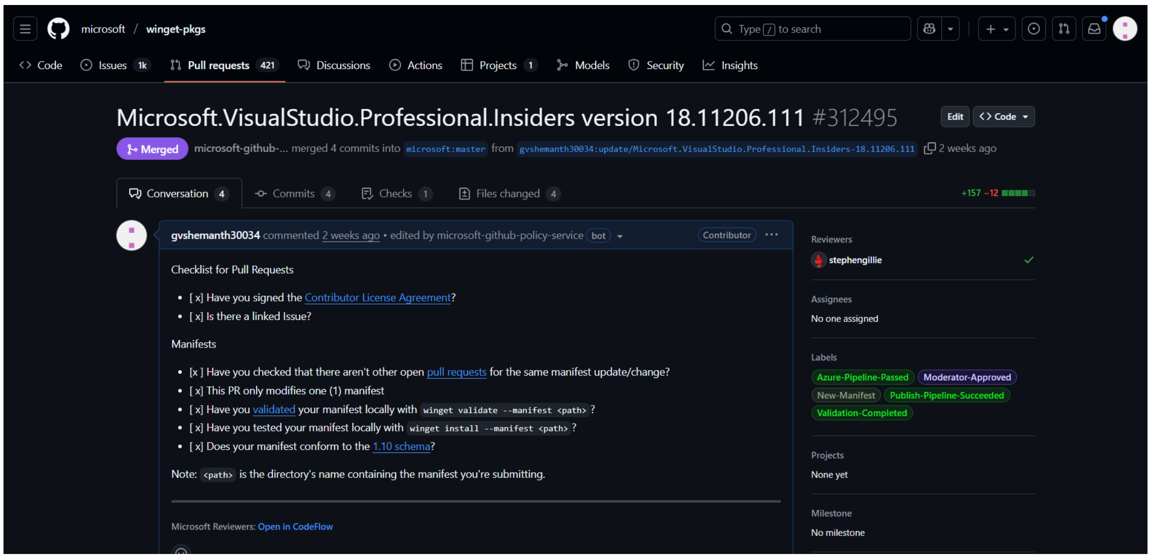
Issue Description

The issue was related to updating the Microsoft.VisualStudio.Professional.Insiders package to the new version 18.11206.111 in Winget. A problem was detected during validation because the installer URL in the manifest was no longer valid. The download link returned a Bad Request (HTTP 400) error, meaning Winget could not access the installer. Because of this, the manifest required a corrected download URL and a new SHA256 hash to match the latest installer from the official Visual Studio source.

Changes Implemented

- Fixed the Broken Installer URL
- Updated the SHA256 Hash
- Revalidated the Manifest Locally

- Ensured Manifest Matches Schema 1.10



Microsoft.VisualStudio.Professional.Insiders version 18.11206.111 #312495

microsoft-github-... merged 4 commits into `microsoft:master` from `gvshemanth30034:update/Microsoft.VisualStudio.Professional.Insiders-18.11206.111` 2 weeks ago

gvshemanth30034 commented 2 weeks ago • edited by microsoft-github-policy-service

Checklist for Pull Requests

- [x] Have you signed the Contributor License Agreement?
- [x] Is there a linked Issue?

Manifests

- [x] Have you checked that there aren't other open pull requests for the same manifest update/change?
- [x] This PR only modifies one (1) manifest
- [x] Have you validated your manifest locally with `winget validate --manifest <path>`?
- [x] Have you tested your manifest locally with `winget install --manifest <path>`?
- [x] Does your manifest conform to the 1.10 schema?

Note: `<path>` is the directory's name containing the manifest you're submitting.

Microsoft Reviewers: Open in CodeFlow

Reviewers
stephengillie ✓

Assignees
No one assigned

Labels
Azure-Pipeline-Passed
Moderator-Approved
New-Manifest
Publish-Pipeline-Succeeded
Validation-Completed

Projects
None yet

Milestone
No milestone

PR 3 – Microsoft winget-pkgs (#310578)

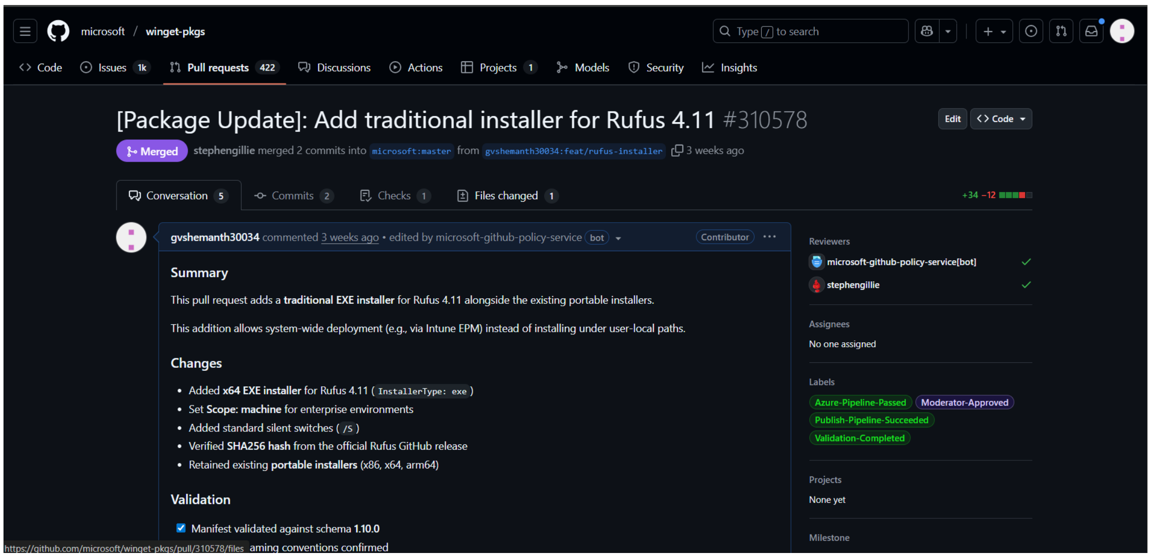
Repository: Microsoft / winget-pkgs **Status: Merged Link:** `https://github.com/microsoft/winget-pkgs/pull/310578`

Issue Description

The issue requested adding a traditional EXE installer for Rufus 4.11, since only portable versions were available. This update was needed to support system-wide installation in enterprise environments. The manifest required a new installer URL, correct SHA256 hash, and proper schema validation.

Changes Implemented

- Added the x64 traditional EXE installer for Rufus 4.11.
- Set Scope: machine to support system-wide deployment.
- Added silent install switches (/S) for automated installation.



[Package Update]: Add traditional installer for Rufus 4.11 #310578

stephengillie merged 2 commits into `microsoft:master` from `gvshemanth30034:feat/rufus-installer` 3 weeks ago

gvshemanth30034 commented 3 weeks ago • edited by microsoft-github-policy-service bot

Summary

This pull request adds a **traditional EXE installer** for Rufus 4.11 alongside the existing portable installers.

This addition allows system-wide deployment (e.g., via Intune EPM) instead of installing under user-local paths.

Changes

- Added **x64 EXE installer** for Rufus 4.11 (`InstallerType: exe`)
- Set **Scope: machine** for enterprise environments
- Added standard silent switches (`/S`)
- Verified **SHA256 hash** from the official Rufus GitHub release
- Retained existing **portable installers** (x86, x64, arm64)

Validation

- [x] Manifest validated against schema **1.10.0**

https://github.com/microsoft/winget-pkgs/pull/310578/files aming conventions confirmed

Reviewers
microsoft-github-policy-service[bot] ✓
stephengillie ✓

Assignees
No one assigned

Labels
Azure-Pipeline-Passed
Moderator-Approved
Publish-Pipeline-Succeeded
Validation-Completed

Projects
None yet

Milestone

PR 4 – First Contributions (#106339)

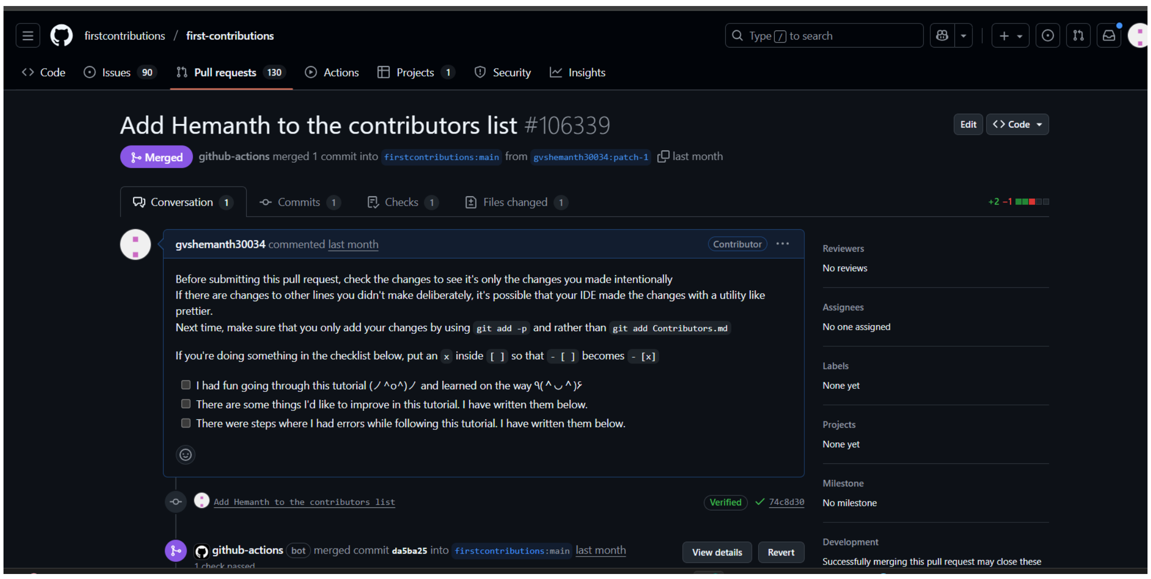
Repository: First Contributions **Status: Merged Link:** `https://github.com/firstcontributions/first-contributions/pull/106339`

Issue Description

The issue was about adding my name to the Contributors.md file in the FirstContributions repository. This repo helps beginners learn how to make their first pull request. I added my name to the contributors list as part of completing the beginner contribution tutorial.

Changes Implemented

- Added my name (Hemanth) to the Contributors.md file
- Made sure only the required line was added and no other content was modified.


firstcontributions / first-contributions

Code | Issues 90 | Pull requests 130 | Actions | Projects 1 | Security | Insights

Add Hemanth to the contributors list #106339

Merged github-actions merged 1 commit into `firstcontributions:main` from `gvshemanth30034:patch-1` last month

Conversation 1 | Commits 1 | Checks 1 | Files changed 1 | +2 −1

gvshemanth30034 commented last month — Contributor

Before submitting this pull request, check the changes to see it's only the changes you made intentionally
If there are changes to other lines you didn't make deliberately, it's possible that your IDE made the changes with a utility like prettier.
Next time, make sure that you only add your changes by using `git add -p` and rather than `git add Contributors.md`

If you're doing something in the checklist below, put an `x` inside `[ ]` so that `- [ ]` becomes `- [x]`

- [ ] I had fun going through this tutorial (／^o^)／ and learned on the way ٩(^◡^)۶
- [ ] There are some things I'd like to improve in this tutorial. I have written them below.
- [ ] There were steps where I had errors while following this tutorial. I have written them below.

`Add Hemanth to the contributors list` Verified ✓ 74c8d30

github-actions bot merged commit **da5ba25** into `firstcontributions:main` last month
1 check passed

View details | Revert

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

PR 5 – KLGLUG / Y24 Open Source Engineering (#175)

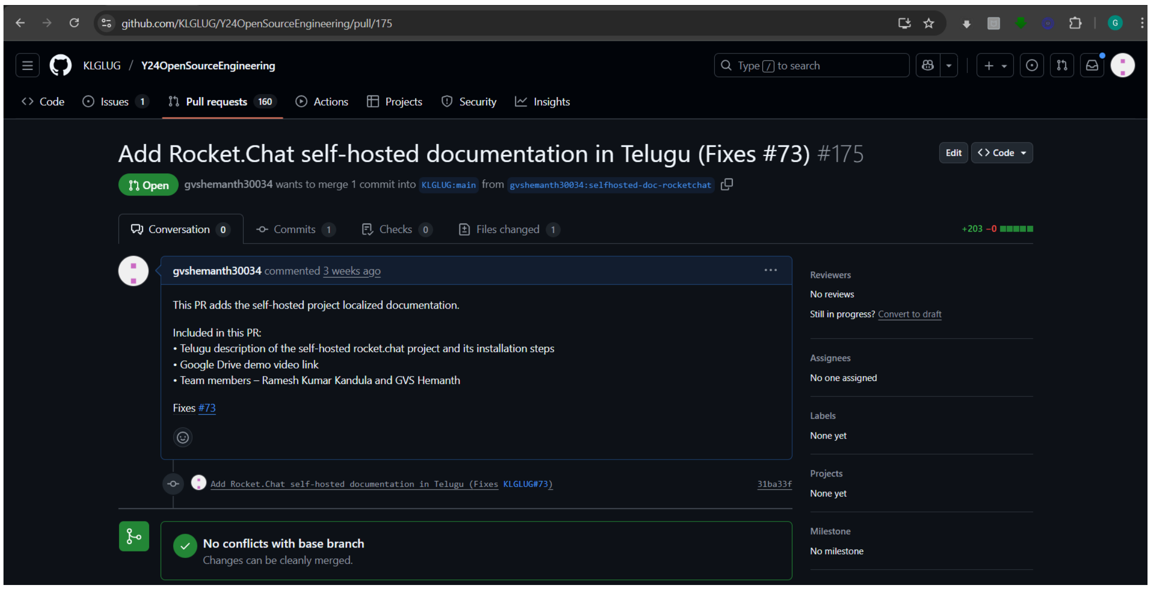
Repository: KLGLUG / Y24OpenSourceEngineering **Status: Under Review (Faculty counts as "Accepted") Link:** `https://github.com/KLGLUG/Y24OpenSourceEngineering/pull/175`

Issue Description

Course documentation needed a localized Telugu version and clear instructions for the self-hosted Rocket.Chat project.

Changes Implemented

- Added Telugu translation of the self-hosted Rocket.Chat guide.
- Included Google Drive demo video link.
- Added team details and improved formatting.



github.com/KLGLUG/Y24OpenSourceEngineering/pull/175
KLGLUG / Y24OpenSourceEngineering
Code
Issues 1
Pull requests 160
Actions
Projects
Security
Insights
Edit
Code

Add Rocket.Chat self-hosted documentation in Telugu (Fixes #73) #175

Open gvshemanth30034 wants to merge 1 commit into KLGLUG:main from gvshemanth30034:selfhosted-doc-rocketchat

Conversation 0 · Commits 1 · Checks 0 · Files changed 1 · +203 −0

gvshemanth30034 commented 3 weeks ago

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 – Ramesh Kumar Kandula and GVS Hemanth

Fixes #73

Add Rocket.Chat self-hosted documentation in Telugu (Fixes KLGLUG#73) 31ba33f

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

Summary

These fic 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. Social platforms such as LinkedIn play an important role in demonstrating practical skills, building a developer profile, and engaging with open-source communities.

Each post reflects a key stage of the course: self-hosting, pull requests, and documentation of ongoing learning.

Post 1 – Self-Hosted Rocket.Chat Deployment

Link: `https://www.linkedin.com/pulse/our-self-hosted-rocketchat-proje`
`?trackingId=oqnn4DHJHy3BvZzN0O2%2FMA%3D%3D`

Summary of the Post

This post explains the complete deployment of a self-hosted Rocket.Chat server on Ubuntu 24.04 LTS. It includes installation using Snap, configuration setup, and secure public exposure using ngrok. The post highlights:

- Importance of self-hosting in open-source engineering.
- Hands-on experience with server deployment.
- Real-world communication tools used in enterprise environments.

Post 2 – Entering the World of Open Source Engineering

Link: https://www.linkedin.com/posts/hemanth-gvs-500006348_sharing-my-journey-of-how-an-open-source-activity-7399319684317356032-utm_source=social_share_send&utm_medium=member_desktop_web&rcm=ACoAAFbaq1ABnLzWFBdkW9PhuUe-d-2E51L-KnU

Summary of the Post

This post introduces the beginning of my Open Source Engineering journey and highlights my initial goals:

- Learning GPG, self-hosting, privacy tools, and Linux.
- Making real pull requests on GitHub.
- Understanding open-source licenses, collaboration, and documentation.

It demonstrates motivation, initial planning, and commitment to open-source learning.

Post 3 – My Open-Source Contribution Highlights

Link: https://www.linkedin.com/pulse/im-excited-share-several-my-open?trackingId=BsHsTIhVx6OnjHI3ESDcHQ%3D%3D

Summary of the Post

This post talks about my first GitHub pull request that got merged in the course. It explains:

- How I started contributing to open-source.
- What I learned while using Git, forks, branches, commits, and pull requests.
- How the successful merge motivated me to keep contributing.

Overall Importance

Posting regularly on LinkedIn helped to:

- Build a professional portfolio.
- Showcase technical and collaborative skills.
- Engage with peers, faculty, and developers worldwide.
- Demonstrate continuous learning in open-source engineering.

All three posts reflect clear progress through the course—from learning, to self-hosting, to contributing to major projects such as firstcontributions and Microsoft winget-pkgs.