Open Source Software

Student Details

Name:	MUNJETI JAYANTH KRISHNA
Roll Number:	2400032115
Department:	Computer Science and Engineering
University:	KL University
Course code :	24CS02EF Open Source Software
Semester:	ODD SEMESTER

Submitted to:

DrSripath Roy
Department of Computer Science and Engineering
KL University

Contents

1 Linux Distribution

1.1 Distribution Used: Ubuntu 22.04 LTS

For this project, I have used **Ubuntu 22.04 LTS** as my primary operating system.

1.2 Why Ubuntu?

Ubuntu is one of the most popular Linux distributions for several reasons:

- **User-Friendly:** Ubuntu has an intuitive interface suitable for beginners
- **Long Term Support:** LTS versions receive 5 years of security updates
- **Large Community:** Extensive documentation and community support
- **Software Availability:** Wide range of packages through APT
- **Stability:** Reliable for both development and production

1.3 Key Features of Ubuntu 22.04 LTS

1. **Desktop Environment:** GNOME 42
2. **Kernel Version:** Linux 5.15 LTS
3. **Package Manager:** APT (Advanced Package Tool)
4. **Default Applications:** Firefox, LibreOffice, GNOME utilities
5. **Snap Support:** Built-in support for snap packages

1.4 System Specifications

My system configuration:

- Operating System: Ubuntu 22.04 LTS
- Architecture: x86_64
- Desktop Environment: GNOME
- Shell: Bash 5.1

1.5 Installation Process

The installation involved:

1. Downloaded Ubuntu 22.04 LTS ISO from official website
2. Created bootable USB using Rufus/Etcher
3. Configured dual boot with existing OS
4. Installed essential development tools
5. Configured system for open source development

2 Encryption and GPG

2.1 What is Encryption?

Encryption is the process of converting plaintext into ciphertext to protect data confidentiality. It ensures that only authorized parties can access the information.

2.2 Types of Encryption

2.2.1 Symmetric Encryption

Uses the same key for encryption and decryption. Examples: AES, DES.

2.2.2 Asymmetric Encryption

Uses a public-private key pair. Examples: RSA, ECC.

2.3 GNU Privacy Guard (GPG)

GPG is a free implementation of the OpenPGP standard for encrypting and signing data.

2.4 Installing GPG

```
sudo apt update
sudo apt install gnupg
gpg --version
```

2.5 Generating GPG Keys

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

Steps followed:

1. Selected RSA and RSA (default)
2. Key size: 4096 bits

3. Key validity: 1 year
4. Entered name and email
5. Created strong passphrase

2.6 Listing Keys

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

2.7 Exporting Public Key

```
gpg --armor --export your-email@example.com > public-key.asc
```

2.8 Encrypting Files

```
gpg --encrypt --recipient your-email@example.com document.txt
```

2.9 Decrypting Files

```
gpg --decrypt document.txt.gpg > document.txt
```

3 Sending Encrypted Email

3.1 Email Encryption Overview

Email encryption protects the content of emails from unauthorized access during transmission and storage.

3.2 Tools Used

- **Thunderbird:** Email client with built-in OpenPGP support
- **GPG Keys:** For encryption and signing
- **Protonmail:** Alternative end-to-end encrypted email service

3.3 Setting up Thunderbird with GPG

3.3.1 Installation

```
sudo apt install thunderbird
```

3.3.2 Configuring OpenPGP

Steps followed:

1. Open Thunderbird
2. Go to Account Settings
3. Select End-to-End Encryption
4. Add existing GPG key or generate new one
5. Import recipient's public key

3.4 Sending Encrypted Email

Process:

1. Compose new email
2. Click on Security button
3. Select "Require Encryption"
4. Optionally add digital signature
5. Send email

3.5 Receiving Encrypted Email

When receiving:

1. Email appears encrypted
2. Thunderbird automatically detects encryption
3. Enter GPG passphrase
4. Email content is decrypted and displayed

3.6 Best Practices

- Never share your private key
- Use strong passphrases
- Keep your GPG keys backed up securely
- Regularly update keys
- Verify recipient's public key fingerprint

4 Privacy Tools from prism-break.org

4.1 What is PRISM-Break?

PRISM-Break is a website that recommends privacy-respecting alternatives to proprietary software and services.

4.2 Tool 1: Signal - Encrypted Messaging

Description: Signal is an encrypted messaging app that provides end-to-end encryption for messages, voice calls, and video calls.

Key Features:

- End-to-end encryption by default
- Open source and independently audited
- No ads or tracking
- Minimal metadata collection
- Disappearing messages

Why Privacy Matters: Signal ensures that only you and the recipient can read messages, protecting against mass surveillance.

4.3 Tool 2: Firefox - Web Browser

Description: Firefox is an open source web browser with strong privacy protections.

Privacy Features:

- Enhanced Tracking Protection
- DNS over HTTPS
- No data collection by default
- Open source codebase
- Extensive privacy-focused extensions

Configuration Tips:

- Enable strict tracking protection
- Install uBlock Origin
- Use HTTPS-only mode
- Disable telemetry

4.4 Tool 3: ProtonMail - Encrypted Email

Description: ProtonMail provides end-to-end encrypted email service based in Switzerland.

Key Features:

- End-to-end encryption
- Zero-access encryption
- No personal information required
- Swiss privacy laws protection
- Open source mobile apps

Use Cases:

- Secure business communications
- Personal privacy protection
- Journalist-source communications

4.5 Tool 4: Tor Browser - Anonymous Browsing

Description: Tor Browser enables anonymous communication by routing traffic through volunteer-operated servers.

How It Works:

- Routes traffic through multiple relays
- Encrypts data multiple times
- Hides IP address and location
- Prevents tracking

Best Use Cases:

- Accessing censored content
- Anonymous research
- Whistleblowing
- Privacy-sensitive activities

4.6 Tool 5: VeraCrypt - Disk Encryption

Description: VeraCrypt is a free open source disk encryption software.
Features:

- Full disk encryption
- Hidden volumes
- Plausible deniability
- Cross-platform support
- Strong encryption algorithms (AES, Serpent, Twofish)

Use Cases:

- Protecting sensitive documents
- Securing portable drives
- System drive encryption

5 Open Source License

5.1 License Used: MIT License

For my open source contributions and projects, I primarily work with the **MIT License**.

5.2 What is the MIT License?

The MIT License is a permissive free software license that allows users to:

- Use the software commercially
- Modify the software
- Distribute the software
- Use the software privately
- Sublicense the software

5.3 MIT License Text

```
MIT License

Copyright (c) 2025 Krishna Medapati

Permission is hereby granted, free of charge, to any person
obtaining a copy of this software and associated documentation
files (the "Software"), to deal in the Software without
restriction, including without limitation the rights to use,
copy, modify, merge, publish, distribute, sublicense, and/or
sell copies of the Software, and to permit persons to whom the
Software is furnished to do so, subject to the following
conditions:

The above copyright notice and this permission notice shall be
included in all copies or substantial portions of the Software.

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

5.4 Why Choose MIT License?

1. **Simple and Easy:** Short and easy to understand
2. **Permissive:** Minimal restrictions on reuse
3. **Business-Friendly:** Can be used in proprietary software
4. **Popular:** Widely used and recognized
5. **Compatible:** Works well with other licenses

5.5 Other Common Open Source Licenses

5.5.1 GPL (GNU General Public License)

- Copyleft license
- Requires derivative works to be open source
- Used by Linux kernel

5.5.2 Apache License 2.0

- Permissive like MIT
- Includes patent grant
- Used by Apache projects

5.5.3 BSD License

- Very permissive
- Similar to MIT
- Used by FreeBSD

6 AdGuard Home: Self-Hosted DNS Ad Blocker

6.1 What is AdGuard Home?

AdGuard Home is a network-level, self-hosted DNS ad- and tracker-blocking server. It:

- Blocks ads, trackers, and malware at DNS level for every device on your network
- Provides DNS rewrites and local DNS records (handy for `search.example.com` → internal IP)
- Offers a web UI to manage filters, whitelists, clients and upstream DNS
- Can run as a standalone binary, systemd service, or inside Docker
- Is a straightforward alternative to Pi-hole

6.2 Why use AdGuard Home with SearXNG?

1. Let every device benefit from ad/tracker blocking while your SearXNG instance continues to fetch results.
2. Use DNS rewrites so LAN clients resolve your public SearXNG domain to its internal IP (avoids hairpin/NAT issues).
3. Whitelist search engine domains if any are blocked so SearXNG result fetching keeps working.
4. Centralized DNS logs make troubleshooting easy when SearXNG cannot reach an engine.

6.3 Supported platforms

This guide focuses on Ubuntu 22.04 LTS (works the same on other Debian/Ubuntu family releases). Two deployment options are shown: **binary + systemd** and **Docker Compose**. Pick one.

6.4 Installation — Option A: Binary + systemd (recommended for single server)

Step 1: Update system

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

Step 2: Download AdGuard Home (latest stable) and extract

```
cd /opt
# change the URL if you have a newer tarball; here we use a
   variable for convenience
AG_VER="AdGuardHome_linux_amd64.tar.gz"
# Download tarball (replace with actual latest if needed)
```

```
sudo curl -sL -o /tmp/$AG_VER "https://static.adguard.com/
   adguardhome/release/AdGuardHome_linux_amd64.tar.gz"
sudo tar -xzf /tmp/$AG_VER -C /opt
sudo mv /opt/AdGuardHome /opt/adguardhome
```

Step 3: Create a dedicated user and set ownership

```
sudo useradd --system --no-create-home --group nogroup adguard
sudo chown -R adguard:nogroup /opt/adguardhome
```

Step 4: Create systemd service

```
sudo tee /etc/systemd/system/adguardhome.service > /dev/null <<'
   EOF'
[Unit]
Description=AdGuard Home DNS server
After=network.target

[Service]
Type=simple
User=adguard
Group=nogroup
WorkingDirectory=/opt/adguardhome
ExecStart=/opt/adguardhome/AdGuardHome -c /opt/adguardhome/
   AdGuardHome.yaml -w /opt/adguardhome
Restart=on-failure
RestartSec=5s
LimitNOFILE=65536

[Install]
WantedBy=multi-user.target
EOF
```

Step 5: Start and enable service

```
sudo systemctl daemon-reload
sudo systemctl enable --now adguardhome
sudo systemctl status adguardhome
```

By default, AdGuard Home listens on:

- DNS (UDP/TCP): port 53
- Web UI: port 3000 (HTTP) or 443 if configured with built-in TLS

6.5 Installation — Option B: Docker Compose (if you prefer containerized)

Create a `docker-compose.yml` (example):

```
version: "3.8"
services:
  adguard:
    image: adguard/adguardhome:latest
    container_name: adguardhome
    restart: unless-stopped
    network_mode: "host"     # host network recommended for DNS
       on port 53
    volumes:
      - /opt/adguardhome/workdir:/opt/adguardhome/workdir
      - /opt/adguardhome/conf:/opt/adguardhome/conf
```

Then:

```
sudo mkdir -p /opt/adguardhome/{workdir,conf}
sudo chown -R 1000:1000 /opt/adguardhome
sudo docker compose up -d
```

6.6 Initial web UI setup

Open the AdGuard Home web installer (default): `http://<server-ip>:3000` and follow the wizard:

- Choose admin username password.
- Choose listening interfaces: usually `0.0.0.0:53` (all IPv4) and :::53 for IPv6 if needed.
- Choose web UI port (3000) or enable TLS (recommended to secure UI).
- Save initial configuration.

6.7 Key configuration files and snippets

AdGuard main config — `/opt/adguardhome/AdGuardHome.yaml`

Below is an illustrative minimal config that sets upstreams, DNS rewrites, and access control. If you installed via installer, the YAML will exist; you can edit it after stopping the service and restart afterward.

```
bind_host: 0.0.0.0
bind_port: 53
dns:
  # Upstream DNS servers (order matters)
  upstream_dns:
    - "https://cloudflare-dns.com/dns-query"   # DoH upstream
    - "1.1.1.1"                                # fallback plain
       UDP/TCP
    - "9.9.9.9"
  bootstrap_dns:
    - "1.1.1.1"
  dns64:
    enabled: false

http:
  listen_addr: ":3000"
  # enable: true  # controlled by installer
  # To use built-in HTTPS, configure certs here (optional)

tls:
  enabled: false

clients:
  # allow all by default; use web UI for client groups if needed

filters:
  enabled: true
  # filter rules will be managed from the web UI (blocklists,
     custom rules)

rewrite:
  # local DNS rewrites (useful for SearXNG internal resolution)
  "search.example.com": "192.168.1.20"  # change to your SearXNG
     LAN IP

access_control:
  # allow queries from local network only (example)
  - action: allow
    network: 192.168.1.0/24
  - action: allow
    network: 127.0.0.1/32
  # Deny others by default (managed by web UI)
```

Important: If you use the web UI to change settings, the YAML will be updated automatically. Prefer the UI for filter management and rewrites.

6.8 Integrating AdGuard Home with SearXNG (practical steps)

1. **Add DNS rewrite**: In AdGuard Home web UI → `Settings` → `DNS rewrites` add:

   - `192.168.1.20` (your SearXNG server internal IP)

   This ensures LAN devices resolve your public domain to the internal host instead of going out and back in.

2. **Whitelist search engine domains if needed**: If SearXNG result aggregation breaks because some upstream engine domains are blocked, add those domains to the AdGuard whitelist.

   - Common candidates to whitelist (examples — add only if you observe breakage): `google.com`, `bing.com`, `duckduckgo.com`, `yahoo.com`, `yahooapis.com`

3. **Ensure SearXNG server uses AdGuard for DNS**: On the SearXNG host, set system DNS to point to the AdGuard instance (if AdGuard runs locally, use `127.0.0.1`).

6.9 Make the OS use AdGuard Home as resolver

If AdGuard Home runs on the same host, configure `systemd-resolved`:

```
sudo sed -i '/^\\[Resolve\\]/a␣DNS=127.0.0.1' /etc/systemd/resolved.conf
sudo systemctl restart systemd-resolved
# Verify
resolvectl status
dig @127.0.0.1 search.example.com +short
```

If AdGuard runs on a different host (e.g., `192.168.1.10`), replace `127.0.0.1` with that IP.

6.10 Optional: Reverse proxy the AdGuard Web UI via Nginx with HTTPS

If you want `https://adguard.example.com` for the UI and to use Let's Encrypt certs, use an Nginx site like:

```
server {
    listen 80;
    server_name adguard.example.com;
    location /.well-known/acme-challenge/ { root /var/www/html; }
    location / { return 301 https://$host$request_uri; }
}

server {
    listen 443 ssl http2;
    server_name adguard.example.com;

    ssl_certificate /etc/letsencrypt/live/adguard.example.com/fullchain.pem;
    ssl_certificate_key /etc/letsencrypt/live/adguard.example.com/privkey.pem;

    location / {
        proxy_pass http://127.0.0.1:3000;
        proxy_set_header Host $host;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
        proxy_set_header X-Forwarded-Proto $scheme;
    }
}
```

Then obtain certificate:

```
sudo ln -s /etc/nginx/sites-available/adguard /etc/nginx/sites-enabled/
sudo nginx -t
sudo systemctl reload nginx
sudo certbot --nginx -d adguard.example.com
```

6.11 Firewall checklist

- Allow DNS ports on AdGuard host (if AdGuard is the network DNS):

```
sudo ufw allow 53/tcp
sudo ufw allow 53/udp
```

- Allow web UI access (if you want remote UI):

```
sudo ufw allow 3000/tcp   # or 443/tcp if proxied by nginx
    and using HTTPS
```

- If using SearXNG on same host, allow 80/443 as for SearXNG reverse proxy.

6.12 Common troubleshooting

- **SearXNG can't fetch results:** Check AdGuard query log for blocked domains and whitelist them.
- **Local clients don't resolve public domain to LAN IP:** Ensure DNS rewrite exists and clients use AdGuard as DNS server.
- **AdGuard fails to bind to port 53:** Port 53 may be used by systemd-resolved — either stop/disable resolved (careful), or configure AdGuard to bind to a specific interface. Example to stop resolved:

```
sudo systemctl disable --now systemd-resolved
sudo rm /etc/resolv.conf
sudo bash -c 'echo␣"nameserver␣127.0.0.1"␣>␣/etc/resolv.conf'
```

  Only do this if you understand the implications.

- **DNS leaks or external clients not using AdGuard:** Ensure DHCP handed out AdGuard IP as DNS or set router DNS to use AdGuard.

6.13 Minimal copy/paste setup script (binary + systemd)

```
#!/usr/bin/env bash
set -e
# Variables - edit as needed
ADG_DIR="/opt/adguardhome"
AG_URL="https://static.adguard.com/adguardhome/release/
    AdGuardHome_linux_amd64.tar.gz"
# 1. Update
sudo apt update && sudo apt upgrade -y
sudo apt install curl -y
# 2. Download & extract
sudo mkdir -p $ADG_DIR
sudo curl -sL -o /tmp/adguard.tar.gz "$AG_URL"
sudo tar -xzf /tmp/adguard.tar.gz -C /tmp
sudo mv /tmp/AdGuardHome $ADG_DIR
# 3. Create user and chown
```

```
sudo useradd --system --no-create-home --group nogroup adguard || true
sudo chown -R adguard:nogroup $ADG_DIR
# 4. systemd unit
sudo tee /etc/systemd/system/adguardhome.service > /dev/null <<'EOF'
[Unit]
Description=AdGuard Home DNS server
After=network.target

[Service]
Type=simple
User=adguard
Group=nogroup
WorkingDirectory=/opt/adguardhome
ExecStart=/opt/adguardhome/AdGuardHome -c /opt/adguardhome/AdGuardHome.yaml -w /opt/adguardhome
Restart=on-failure
RestartSec=5s
LimitNOFILE=65536

[Install]
WantedBy=multi-user.target
EOF
# 5. Start service
sudo systemctl daemon-reload
sudo systemctl enable --now adguardhome
echo "AdGuard Home installed. Open http://$(hostname -I | awk '{print $1}'):3000 to finish setup."
```

6.14 SearXNG-specific checklist to run together with AdGuard Home

1. Keep `searxng` bound to `127.0.0.1:8888` and reverse proxy via Nginx (HTTPS) as in your SearXNG config.
2. In AdGuard Home, add a DNS rewrite: `search.example.com -> 192.168.1.20`.
3. Ensure the SearXNG host uses AdGuard as its resolver so outgoing requests follow your network DNS rules.
4. Monitor AdGuard query logs for blocked upstream domains used by SearXNG; whitelist them when necessary.

6.15 Notes and best practices

- Use HTTPS for both SearXNG and AdGuard Admin UI (either built-in or via Nginx + certbot).
- Keep blocklists curated — overly aggressive lists can break sites and APIs SearXNG relies on.

- Use client groups in AdGuard for device-specific policies (e.g., more strict for IoT).
- Keep periodic backups of `/opt/adguardhome/` (configs + filters) and of your SearXNG settings.

6.16 If you want

I can also:

- Produce a single combined shell script that installs AdGuard Home and sets the recommended DNS rewrite for `search.example.com`.
- Produce a Docker Compose stack that runs AdGuard + SearXNG together (if you prefer containers).

7 Top 5 Open Source Contributions

This section highlights the five most impactful and notable pull requests selected from my open source contributions (based on the screenshots and project relevance).

7.1 PR 1: Dashy - Issue Solved (Merged)

Repository: 2400032210/Dashy
PR Number: 2
Status: Merged (3 weeks ago)
Tasks Completed:
Tasks Completed: 6
Comments: 1 discussion

7.1.1 Summary

Resolved a reported issue in the Dashy project by identifying the root cause and applying the correct fix. Completed all 6 tasks associated with the issue, leading to a successful merge. This contribution improved the functionality and stability of the repository.

7.2 PR 2: Add Name to Code Contributions

Repository: Roshanjossey/code-contributions
Status: Open
PR Number: 749

7.2.1 Summary

Submitted an update adding my contributor profile to the project. This repository is widely used for Git and GitHub learning.

7.3 PR 3: Fix Links in OSSU Computer Science Curriculum

Repository: ossu/computer-science
Status: Open
PR Number: 1372

7.3.1 Summary

Fixed broken links for Part 2 and Part 3 of the Systematic Program Design (SPD) curriculum. Improves navigation for thousands of learners.

7.4 PR 4: Localized Markdown Documentation

Repository: KLGLUG/Y24OpenSourceEngineering
Status: Open
PR Number: 172

7.4.1 Summary

Added a newly localized markdown documentation file for the self-hosted project, improving accessibility for non-English speakers.

7.5 PR 5: Improve Error Message for Parallel Stack Deployments

Repository: serverless/serverless
Status: Merged
PR Number: 13146

7.5.1 Summary

Improved the clarity of error messages shown during parallel stack deployments. Enhances the developer experience for Serverless Framework users.

8 LinkedIn Posts

8.1 Post 1: AdGuard Home + DNS Project

Link: `https://www.linkedin.com/pulse/showcasing-my-project-adguard-home-dlna-integrat`
Summary: Shared details about integrating AdGuard Home with DNS and showcased the technical architecture and implementation.
Key Points:

- **Network privacy and ad-blocking**
- **DNS DNS configuration and network setup capabilities**

- **Project implementation and demo**
- **Practical real-world setup**

8.2 Post 2: Recent Open Source Contributions

Link: `https://www.linkedin.com/pulse/my-recent-open-source-contributions-jayanth-kris`

Summary: Highlighted the series of pull requests made in multiple repositories including Dashy, AdGuard, and community-based projects.

Highlights:

- **Continuous contributions to open source**
- **Pull request workflow and best practices**
- **Collaboration with maintainers**
- **Learning through contributions**

8.3 Post 3: My Open Source Journey

Link: `https://www.linkedin.com/pulse/my-open-source-journey-from-confusion-confidence`

Summary: Discussed the transition from starting open source with confusion to gaining confidence and developing meaningful projects.

Takeaways:

- **Overcoming beginner hesitation**
- **Finding good first issues**
- **Hands-on learning and growth**
- **Consistency in contributing**

9 Conclusion

Open-source participation has been a defining phase of my technical and professional growth. It enabled me to apply theoretical knowledge to real projects, collaborate with maintainers across the world, and contribute to tools that impact real users and developers. Through this journey, I gained valuable experience in problem-solving, version control, debugging, documentation, and collaborative development.

More importantly, open source is not just about writing code—it is about community, learning, and innovation. It taught me how to work in distributed teams, understand complex codebases, and communicate effectively in a professional environment.

This journey has strengthened my confidence as a developer and inspired me to continue contributing, exploring more challenging projects, and helping others begin their open-source journey. I look forward to growing further and being a part of meaningful global software development.