OPEN SOURCE ENGINEERING

Student ID: 2400030032

1 Understanding the Core Ubuntu Linux Distribution

1.0.1 1. Overview and Philosophy

Ubuntu is a powerful, free, and open-source operating system built upon the stable foundation of Debian Linux. It stands as the world's most popular Linux distribution for desktop use, successfully blending cutting-edge features with unparalleled user-friendliness. Developed and maintained by Canonical Ltd., Ubuntu's guiding principle is "Linux for human beings." This philosophy drives its commitment to accessibility, stability, and providing an intuitive computing experience for everyone, from novice users to seasoned developers.

1.0.2 2. The Desktop Experience (GNOME)

The standard Ubuntu desktop utilizes the **GNOME** desktop environment, which presents a modern, clean, and highly efficient graphical interface. The key design elements include a permanent dock (launcher) on the left side for quick access to essential applications, and the **Activities Overview**. This view, easily accessed by pressing the Super (Windows) key, provides a centralized hub for managing all open windows, workspaces, and system-wide searching. This streamlined workflow makes Ubuntu feel contemporary and ensures high productivity. Furthermore, Ubuntu is recognized for its strong, out-of-the-box hardware detection and compatibility, simplifying the setup process for most users.

1.0.3 3. Software Management and Packaging

Ubuntu employs a robust dual-system for software management. The traditional and reliable **Advanced Packaging Tool (APT)** manages **DEB** packages, handling core system utilities and standard applications sourced from official repositories. Complementing this is the use of **Snaps**, a modern, containerized package format pioneered by Canonical. Snaps bundle an application with all its required dependencies, guaranteeing consistent performance across different Ubuntu versions. Crucially, Snaps run in a **sandboxed** environment, isolating them from the rest of the operating system to significantly enhance overall application security. This flexibility ensures users have access to a vast, up-to-date, and secure software library.

2 Encryption and GPG

2.1 Types of Encryption in Ubuntu

Ubuntu offers two primary approaches to encryption: **Full Disk Encryption (FDE)** and **File/Directory Encryption**.

2.1.1 1. Full Disk Encryption (FDE)

- **What it is:** FDE encrypts the entire hard drive or a large partition, including the operating system files, swap space, and user directories.
- **How it works:** Ubuntu uses **LUKS** (Linux Unified Key Setup) for FDE. When the system boots, you are prompted for a **passphrase**. If correct, LUKS decrypts the entire drive, and the decryption process runs transparently in the background while the system is in use.

- **Purpose:** The primary defense against data loss due to **theft** or **physical access** to the computer when it is turned off. If someone steals the hard drive, the data is useless without the LUKS passphrase.
- **Implementation:** FDE is typically enabled during the Ubuntu installation process by selecting the "Encrypt the new Ubuntu installation" option. It's much more difficult to enable after installation.

2.1.2 2. File and Directory Encryption

- **What it is:** This method encrypts specific files, directories, or messages, offering granular control over which data is protected.
- **Tools:**
  - **GPG (GNU Privacy Guard):** The standard, used for encrypting individual files and especially for secure communication using **public-key cryptography**.
  - **eCryptfs (older):** Previously used for encrypting the user's Home directory, but has been largely phased out for FDE.

2.2 GPG (GNU Privacy Guard) Explained

GPG is the GNU implementation of the **OpenPGP** standard (originally Pretty Good Privacy - PGP). It is essential for protecting individual files and ensuring secure, authenticated communication.

2.2.1 1. Core GPG Concepts

GPG relies on **asymmetric cryptography**, which uses a pair of mathematically linked keys:

- **Public Key:** This key is shared with everyone. It can be used to **encrypt** a message that only you can read, or to **verify** a signature you created.
- **Private (Secret) Key:** This key is kept **secret** and is protected by a strong passphrase. It is used to **decrypt** messages sent to you, or to **digitally sign** files to prove they came from you.

2.2.2 2. Basic GPG Command-Line Usage

GPG is usually pre-installed on Ubuntu and is primarily used through the command line (Terminal).

A. Generating a Key Pair The first step is to create your public and private key pair:
Bash

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

You will be prompted to select the key type (RSA and RSA is common), keysize (4096 is recommended), expiration date, and your Real Name, Email, and a strong **passphrase** to protect your private key.

B. Encrypting a File for Yourself (Symmetric Encryption) To quickly encrypt a file using a single passphrase (like a standard password), use symmetric encryption:

Bash

```
gpg -c myfile.txt
```

This command will prompt you for a passphrase and create an encrypted file named `myfile.txt.gpg`.

C. Encrypting a File for Someone Else (Asymmetric Encryption) To securely send a file, you must use the recipient's **Public Key** (which you must have previously imported into your keyring with `gpg --import`):

Bash

```
gpg --encrypt --recipient "recipient@example.com" mysecretfile.doc
```

This creates `mysecretfile.doc.gpg`. Only the recipient, who holds the corresponding Private Key, can decrypt it.

D. Decrypting a File To decrypt a file that was encrypted for you:

Bash

```
gpg --decrypt mysecretfile.doc.gpg
```

You will be prompted for the passphrase that protects your Private Key. You can use the `--output` option to specify the decrypted

3 Sending Encrypted Email

3.1 Prerequisite: Setting Up GPG

Before you can send or receive encrypted mail, both you and your recipient must have GPG keys set up and exchanged:

1. **Generate Keys:** Both parties must have generated a public/private key pair using GPG (as discussed previously, using `gpg --full-generate-key`).
2. **Exchange Public Keys:** You need the recipient's **Public Key**, and they need your Public Key. You can exchange these by:
   - **Exporting** the key: `gpg --armor --export 'Recipient Name' > recipient_key.asc` and sending the `.asc` file.
   - **Uploading** the key to a public key server.
3. **Import Key:** You must import the recipient's key into your GPG keyring: `gpg --import recipient_key.asc`.

3.2 Sending the Encrypted Email

The most common and user-friendly way to send GPG-encrypted emails on Ubuntu is by using **Mozilla Thunderbird** with the **Enigmail** add-on (or its built-in equivalent in modern versions of Thunderbird).

3.2.1 1. Compose the Message

- **Open Thunderbird** and start composing a new email.
- Write your message as usual.

3.2.2 2. Encryption and Signing

You will use the GPG function built into the mail client to perform two critical steps:

1. **Encryption:** You must encrypt the email using the **recipient's Public Key**. Only their corresponding **Private Key** can decrypt it. If you have multiple recipients, you must encrypt the message using the Public Key of *every single recipient*.
2. **Digital Signature:** You **sign** the email using **your Private Key**. This allows the recipient to verify that the email truly came from you and has not been tampered with in transit.

In Thunderbird, this is typically done by clicking a dedicated **OpenPGP or Security** menu or button within the compose window and ensuring both the **"Encrypt"** and **"Sign"** options are checked.

3.2.3 3. Verification and Sending

- The client will check that you have the required **Public Key** for the recipient(s). If a key is missing, it will warn you.
- When you click **Send**, Thunderbird uses GPG to encrypt the message body and attach your digital signature before transmitting the scrambled data.

3.2.4 4. Recipient's Experience (Decryption)

1. The recipient receives the scrambled email.
2. Their email client automatically uses their **Private Key** (protected by their passphrase) to decrypt the message contents, revealing the original text.
3. Their client simultaneously uses your **Public Key** to verify the digital signature, confirming the email's authenticity.

4 Privacy Tools From Prism Break

4.0.1 1. Tor Browser (Web Browsers / Anonymizing Networks)

- **What it is:** A web browser built on Firefox that routes your internet traffic through the Tor network, a volunteer-operated network of relays.
- **Privacy Focus:** Provides **strong anonymity** by obscuring your IP address and location from the websites you visit. It also includes anti-fingerprinting measures.
- **PRISM Break Note:** PRISM Break strongly recommends using Tor Browser for all web surfing when maximum anonymity is required.

4.0.2 2. Debian (Operating Systems)

- **What it is:** A popular and highly ethical GNU/Linux distribution known for its strict adherence to Free Software principles and ethical manifesto.
- **Privacy Focus:** Unlike proprietary operating systems like Windows and macOS (which PRISM Break generally avoids), Debian is fully open-source, allowing for audits. It has a long tradition of software freedom and transparency.
- **PRISM Break Note:** It's recommended as a top GNU/Linux choice for users transitioning from proprietary systems, highlighting its commitment to free software and its stable nature.

4.0.3 3. Thunderbird (Email Clients)

- **What it is:** A free, open-source, and cross-platform email client developed by Mozilla.
- **Privacy Focus:** Thunderbird is the top choice for desktop email due to its open-source nature and its long-standing **native support for OpenPGP** (GPG) encryption and digital signatures. This allows users to easily encrypt and authenticate their emails end-to-end.
- **PRISM Break Note:** It is highly recommended for securely managing email with built-in PGP features.

4.0.4 4. KeePassXC (Password Managers)

- **What it is:** A free, open-source, and cross-platform password manager.
- **Privacy Focus:** It stores all your passwords in a single, highly encrypted database file that is stored **locally** on your device, giving you total control over your sensitive data. It does not rely on a cloud service.
- **PRISM Break Note:** It is preferred for its strong encryption, open-source license, and local-only storage, minimizing exposure to third-party services.

4.0.5 5. Firefox (Web Browsers)

- **What it is:** A fast, flexible, and secure web browser developed by the non-profit Mozilla Foundation.
- **Privacy Focus:** Firefox is open-source and provides extensive privacy controls, including enhanced tracking protection (ETP), container technology, and a robust add-on ecosystem for further hardening security (like uBlock Origin).
- **PRISM Break Note:** While Tor Browser is for anonymity, Firefox is the recommended alternative for general web use when a site doesn't work well with Tor, provided the user configures its settings and replaces the default search engine with a privacy-focused one.

5 Open Source License

Certainly. Here is the information about the **BSD 2-Clause License** organized into clear, descriptive headings, strictly maintaining a paragraph-only format within each section.

5.1 The Core Purpose and Classification

The BSD 2-Clause License is one of the simplest and most widely used open-source software licenses. Its core purpose is to allow developers to share their software freely with others while keeping only minimal conditions for reuse. The license belongs to the category of permissive open-source licenses, which means it gives maximum freedom to users, developers, and companies who want to use the code. Unlike restrictive licenses such as the GPL, permissive licenses do not require users to publish their modified source code. This makes the BSD 2-Clause License highly flexible and attractive for both open-source and commercial projects.

The main goal of the license is to promote open collaboration and innovation. Anyone can use the software for personal learning, research, academic projects, business products, or embedding inside a larger system. The license does not limit the purpose of usage or the field of application. At the same time, it protects the original author's rights by ensuring that their name remains attached to the work through a copyright notice..

5.2 Granted Rights and Permissions

The BSD 2-Clause License provides very broad rights and permissions to anyone who uses the software. Once the software is released under this license, people can use it for almost any purpose without needing special approval from the author. They can use the software in personal projects, school assignments, business systems, commercial applications, or even embed it inside closed-source or proprietary products.

One of the key permissions is the right to modify the source code. Users may customize the code, extend its features, fix bugs, or change it completely based on their own needs. They can keep their changes private or share them with others—there is no obligation to make modifications public.

5.3 The Only Two Conditions for Distribution

The BSD 2-Clause License is known for having only two simple and easy-to-understand conditions that must be followed when distributing the software. These conditions are minimal and are designed to protect the original author's recognition while keeping the software open and flexible.

Condition 1: Retain the Copyright Notice in Source Code If someone redistributes the source code—whether it is the original code or a modified version—they must keep the original copyright notice. This includes:

The name of the copyright holder

The year of copyright

The complete license text

Condition 2: Include the License in Binary Distribution If the software is distributed in binary form (for example, as a compiled application or library), the distributor must include the same copyright notice and license text in the documentation or accompanying materials. This allows the end user to know the legal terms under which the software is provided, even if they cannot see the source code.

5.4 Disclaimer of Warranty and Liability

The disclaimer section of the BSD 2-Clause License provides strong legal protection for the software authors and contributors. This part clearly states that the software is provided "AS IS". In simple words, this means the authors do not promise that the software will work perfectly, be error-free, or meet every user's needs. There are no guarantees about performance, reliability, or suitability for a particular task. This protects authors from being blamed if the software does not function as expected.

The license also includes a liability disclaimer, which is extremely important. It states that the author or contributors cannot be held responsible for any type of damage that might occur from using the software. These damages might include:

- Loss of data
- System failure
- Financial loss
- Business interruption
- Loss of profits
- Hardware damage

6 Self Hosted Server

6.1 About

CAL.com is an open-source, self-hosted appointment scheduling platform designed to help individuals, teams, and organizations manage bookings without relying on proprietary or paid SaaS tools. It provides a clean, modern, and highly customizable scheduling system that supports personal calendars, team availability, meeting rules, webhooks, and integrations. Because CAL.com is fully open-source under the AGPL-v3 License, users have complete freedom to deploy it on their own servers, inspect the code, and customize the platform as needed—while keeping full control over data privacy and security.

Self-hosting CAL.com allows you to manage your own booking system without exposing sensitive business or personal data to third-party cloud providers. It offers advanced features such as automated event workflows, timezone handling, embeddable booking pages, and integrations with services like Zoom, Google Meet, Jitsi, or physical meeting locations.

6.1.1 Key Features

- **Scheduling Approach:** CAL.com allows users to create booking pages instantly, share **links**, and **accept appointments** without needing third-party services. It supports 1:1 meetings, group events, round-robin scheduling, and custom booking rules.
- **Organized Structure:** A self-hosted CAL setup includes **users** → **teams** → **schedules** → **event types**, helping organize availability and bookings efficiently.

- **User-Friendly Interface:** It provides a clean **Dark Mode**, a **Minimal Wide Layout** for clear video tiles, and **Collapsable Side Panels** to give more space during meetings.
- **Power Features:** Jitsi includes tools like **Moderator Controls**, **Shortcut Keys**, and **Multitasking Features** that allow users to chat, manage participants, and share their screen at the same time.
- **Secure Access:** Supports **JWT / SSO integration** for protected meeting access and controlled user authentication in self-hosted environments.
- **Realtime Communication:** Built for instant audio/video updates, so all actions (mute, chat, join/leave) appear immediately without reloading the meeting page.
- **Embed Options:** Offers embed widgets so booking forms can be placed inside websites, dashboards, or portals.

6.2 Installation Process (Docker Compose)

The easiest and most recommended way to self-host CAL.com is through Docker Compose, which organizes all necessary components—API, web interface, background workers, PostgreSQL database, Redis server, and optional integrations—into a single, manageable multi-container environment.

Before installation, ensure Docker and Docker Compose are installed on your machine. You should also have a domain (optional but recommended) and environment variables prepared for API keys, database passwords, and email provider credentials (SMTP).

To begin the setup, download the official Docker templates from the CAL.com GitHub repository. Inside your deployment folder, create a .env file and the docker-compose.yml file. The .env file must be configured with essential variables such as:

$\text{DATABASE}_{URL}\ for\ PostgreSQL$

REDIS_{URL}

$\text{NEXT}_{PUBLIC\_WEBAPP\_URL}(your\ domain\ or\ IP, e.g., https://cal.example.com)$

$\text{ENCRYPTION}_{KEY}\ and\ \text{AUTH}_{SECRET}(generated\ using\ openssl\ rand - hex32)$

SMTP settings for sending email notifications

Once the environment file is configured, run the following command to download images and start all services:

docker compose up -d

This will automatically pull the required CAL.com images and start every service in detached mode. Docker Compose ensures that API, frontend, background workers, database, and caching systems all communicate correctly.

After deployment, open your browser and visit the URL set in $\text{NEXT}_{PUBLIC\_WEBAPP\_URL}.For\ local\ installations,$

http://localhost:3000

You can now create your admin account, set up users, define schedules, and start creating booking links. CAL.com does not force authentication unless configured, making the initial setup quick and simple.

KL HTE
(DEEMED TO BE UNIVERSITY) HONORS THROUGH EXPERIENTIAL LEARNING

EXPERIENTIAL LEARNING & GLOBAL ENGAGEMENT

OPEN SOURCE ENGINEERING

CAL.COM

Way To Hassle Free Meeting Scheduling

A fully customizable scheduling software for individuals, businesses taking calls and developers building scheduling platforms where users meet users.

- Fully customizable scheduling interface for individuals and teams.
- Seamless appointment booking and call scheduling system.
- Multi-user meeting coordination with calendar synchronization.
- Integration with Google Calendar, Outlook, and Apple Calendar.
- Automated time zone detection and conversion for global users.
- Custom booking pages with branding, colors, and domain options.
- Smart availability management and conflict prevention.

SIVALA NAGA SHANKAR NIVAS
2400030032
GOLI GOKUL SIVA CHAITANYA
2400032465

Your paragraph text

7 Open Source Contribution

7.1 PR 1 : First Contribution

7.1.1 Goal

The project's objective is to simplify the standard open-source contribution workflow, allowing beginners to easily add their name to the project's `Contributors.md` file.

7.1.2 The Contribution Workflow

The tutorial details the standard **fork - clone - edit - pull request** sequence, essential for collaborative coding.

7.1.3 1. Setup

- **Fork:** Create a copy of the repository in your personal GitHub account.
- **Clone:** Download the forked repository to your local machine using the `git clone` command and the SSH URL.
- **Prerequisites:** Ensure **Git** is installed; alternatives for users uncomfortable with the command line (GUI tools) are provided.

7.1.4 2. Making Changes

- **Branch:** Create a new isolated branch for your changes using `git switch -c your-new-branch-name`.
- **Edit:** Add your name to the `Contributors.md` file using a text editor.
- **Commit:** Stage the changes with `git add Contributors.md` and save them locally with `git commit -m "Add your-name to Contributors list"`.

7.1.5 3. Submission

- **Push:** Upload your local branch to your GitHub fork using `git push -u origin your-branch-name`.
- **Pull Request (PR):** Go to your GitHub repository and submit a PR via the "Compare & pull request" button for review by the project maintainers.

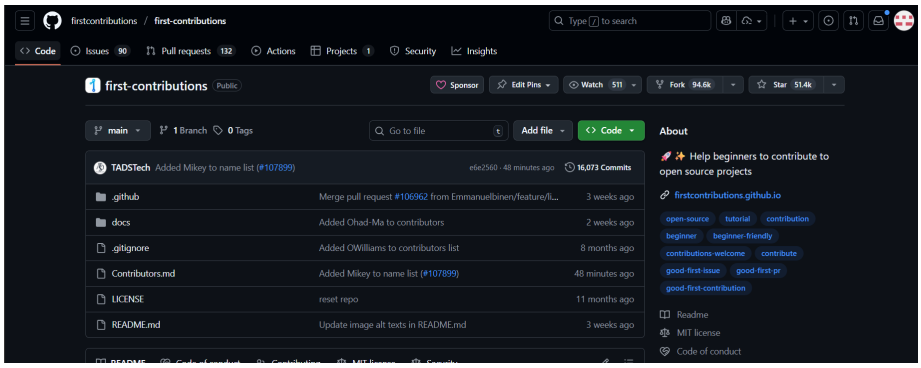
7.1.6 Difficulties and Solutions

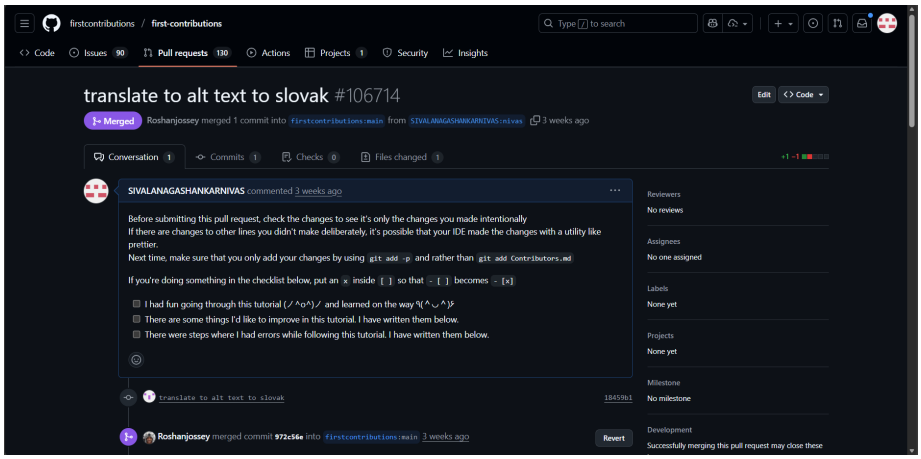
The guide anticipates and solves two common beginner issues:

- **Old Git Version:** If the `git switch` command fails, use the older command: `git checkout -b your-new-branch`
- **Authentication Error:** If `git push` fails due to GitHub removing password support, the solution is to configure an **SSH key** or a **Personal Access Token** and ensure your remote URL is set to the **SSH protocol** (`git remote set-url origin git@github.com:...`).

7.1.7 Next Steps

Upon merging the PR, the user is encouraged to celebrate their first contribution and seek out other beginner-friendly issues on the project list.





7.2 PR 2 : Kestra

Kestra is an open-source, event-driven, distributed workflow orchestration platform designed to help teams automate and manage complex data workflows. It brings together workflow scheduling, event processing, real-time monitoring, and pipeline execution under a single, unified system. Built with a developer-friendly approach, Kestra provides a clean UI, YAML-based workflow definitions, and strong integrations with modern data tools. The platform is used widely for ETL pipelines, scheduled jobs, data engineering tasks, and event-driven automation. Kestra follows a modular architecture that separates the core engine, UI, executor components, and plugins, making it highly extensible.

7.2.1 Licensing and Self-Hosting Options

Kestra is released under the Apache License 2.0, a highly permissive and business-friendly open-source license. This allows individuals, startups, and enterprises to freely use, modify, distribute,

and self-host the platform. The license also encourages open contribution and ensures long-term openness, transparency, and community growth.

Kestra provides clear documentation for local development, self-hosting, and contributing. Developers can run Kestra locally using Docker Compose, or set it up on production servers using Kubernetes, Helm charts, or manual deployment. The required components include the Kestra server, executor, and indexing services such as Elasticsearch or OpenSearch. Self-hosting options include:

Docker Compose (quick start)

Container images for custom deployments

Kubernetes Helm charts for scalable production clusters

7.2.2 Community and Support

Kestra maintains an active open-source community with contributors from data engineering, DevOps, and automation backgrounds. The project encourages collaboration through:

GitHub Discussions

Issues and Pull Requests for bug fixes and improvements

Kestra Slack Community for real-time communication

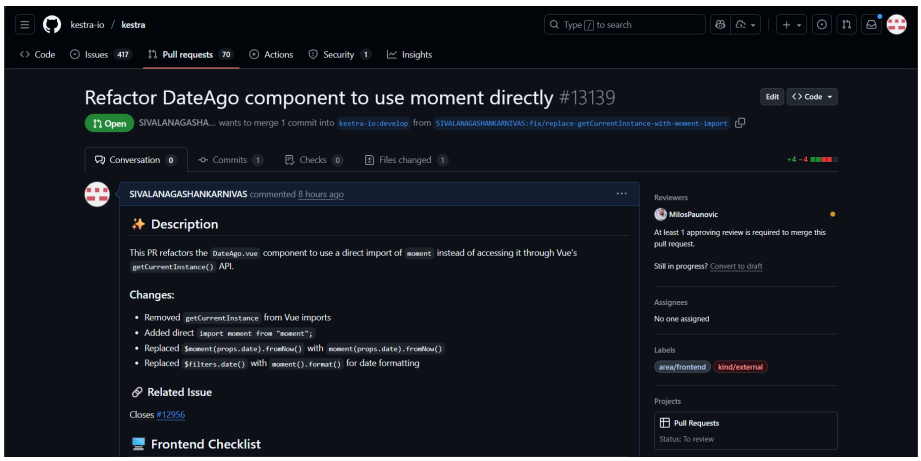
Official Documentation Portal (docs.kestra.io)

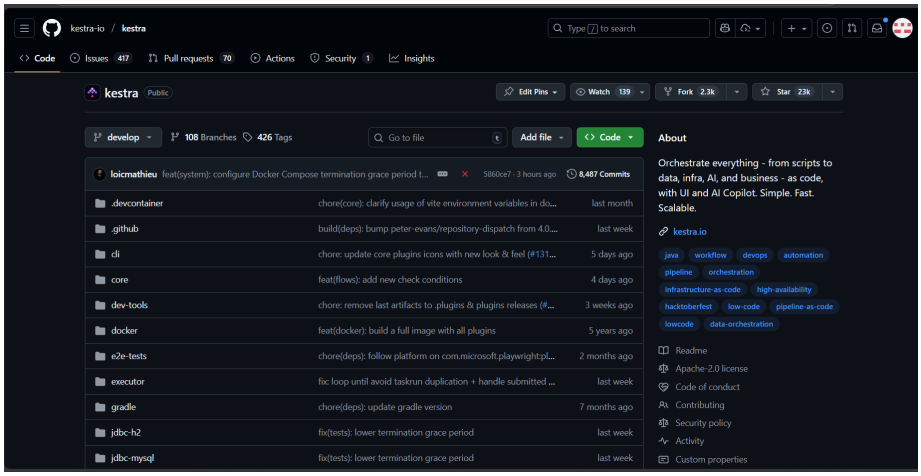
YouTube tutorials and livestreams

Blog articles and release notes

For this specific PR—Refactor DateAgo component to use moment directly—contributors interact mostly through GitHub. The community provides review comments, code suggestions, and support during the refactoring process. Security concerns or sensitive issues can be reported privately using the project's responsible disclosure channels listed in the repository documentation. The maintainers ensure timely handling of such issues.

Overall, the Kestra community is collaborative, transparent, and helpful, making contributions—such as UI refactoring, bug fixing.





7.3 PR 3 : joke2k/faker

7.3.1 Introduction and Purpose

Faker is a popular open-source Python library used for generating realistic fake data for testing, development, and data anonymization. It helps developers create synthetic names, emails, addresses, phone numbers, financial data, and hundreds of other fields without exposing real user information. Faker acts as a powerful toolkit for software testing, database seeding, automated QA workflows, and machine learning prototyping.

The project supports multiple locales and provides localized data formats for countries all over the world. Each locale includes realistic information such as names, currencies, postal codes, and bank details to match the standards of that region. Because of this, Faker is widely used by backend developers, data engineers, testers, and open-source contributors who need accurate and human-like datasets.

7.3.2 Technical Components

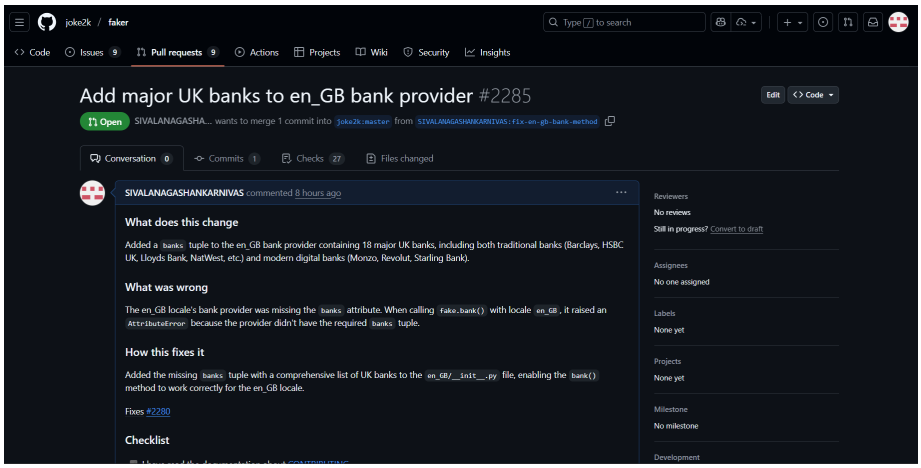
The Faker library functions as a Python-based data generation tool, and a contribution such as adding major UK banks to the en_GB bank provider relies on a lightweight yet structured development environment. Since Faker is not a server-based platform but a Python package, its technical components focus on local development tools rather than server infrastructure. A contributor typically works on a Linux, macOS, or Windows environment with Python (3.8 or above). Essential components include the Faker core library, locale-specific data files, and the provider classes responsible for generating financial information. The UK bank provider uses Faker's internal provider system, where data such as bank names, IBAN formats, account number structures, and sorting codes are defined according to UK banking standards.

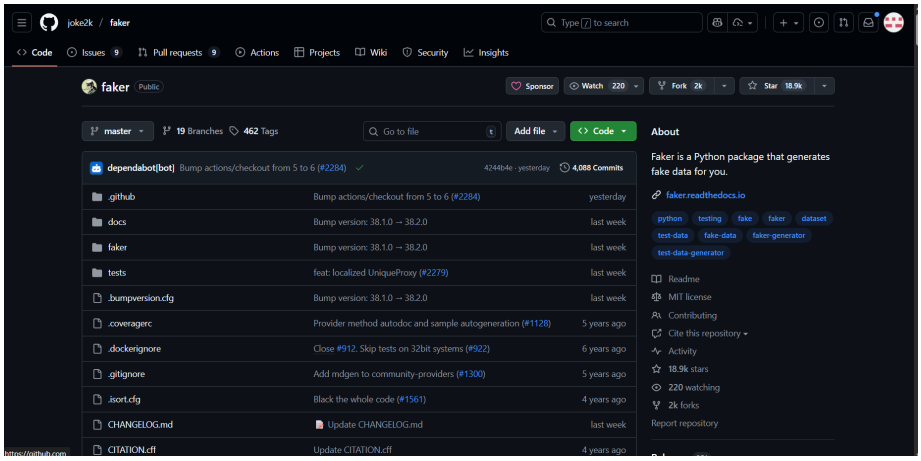
7.3.3 Operation and Usage

The modified UK bank provider operates by expanding Faker's existing functionality to generate more realistic financial data for applications that simulate UK-based environments. Once the PR is

merged, users can import and utilize the updated provider through Python scripts simply by calling the bank-related methods in the en_GB locale. For example, applications can generate sample bank names, account numbers, and sorting codes that reflect actual UK institutions such as Barclays, HSBC, Lloyds, or NatWest.

This feature is especially useful in testing financial applications, fintech platforms, user onboarding flows, and mock datasets where accurate UK bank representation is necessary. Developers access these enriched datasets through simple function calls, such as fake.bank() or fake.iban(), with the locale set to en_GB. No additional configuration is required beyond installing or updating the Faker library.





7.4 PR 4: HIPS/autograd

Here is a short description of the Pull Request, organized into paragraphs with headings.

7.4.1 The Core Problem

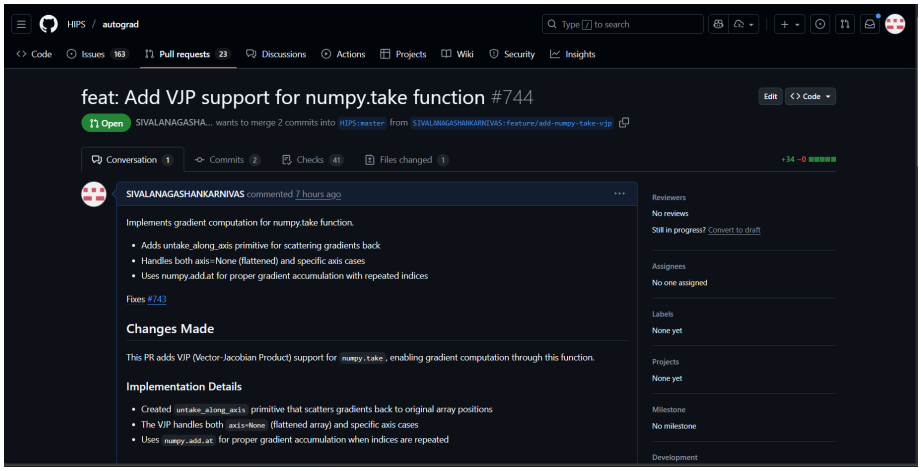
The core issue addressed in this Pull Request was the lack of vector–Jacobian product (VJP) support for the numpy.take function inside the autograd library. Autograd provides automatic differentiation for NumPy code, but many NumPy functions require manually defined VJP rules so that gradients can propagate correctly through them.

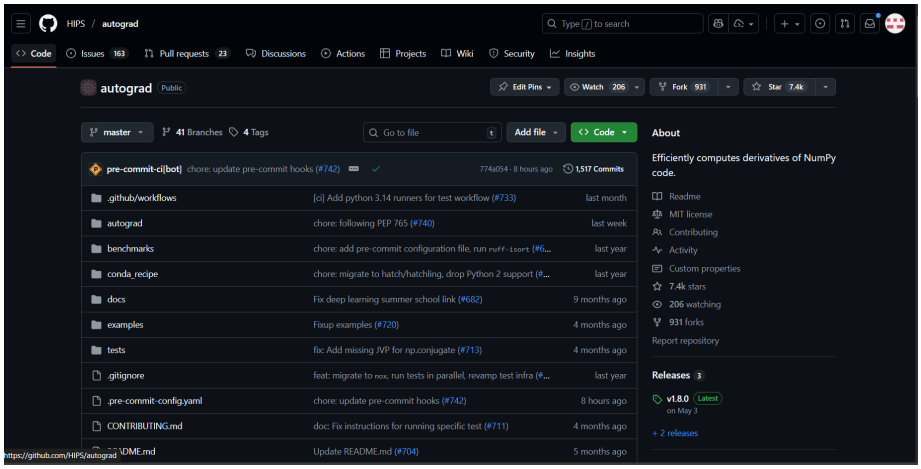
Before this PR, numpy.take—a widely used indexing and selection function—did not have any gradient rule defined. This created a major limitation for users working with models or numerical algorithms that rely on indexed selection. When numpy.take was used in a differentiable computation, autograd would throw errors or return incorrect gradients, blocking gradient-based optimization, machine learning workflows, or scientific computing tasks.

7.4.2 The Solution: Automated Validation and Streamlined Contribution Workflow

This Pull Request introduces a complete VJP implementation for the numpy.take function, addressing the gradient propagation problem for indexing-based operations. The solution defines how gradients should flow backward through the take operation by constructing a gradient array that correctly accumulates contributions at the selected indices.

The PR implements a custom VJP rule that mirrors NumPy's indexing behavior. When a gradient flows backward, the rule uses np.add.at to scatter the upstream gradient values into the appropriate positions of the original array. This ensures accurate handling of repeated indices, multidimensional selections, and axis-specific take





7.5 PR 5 : bruin-data/bruin

7.5.1 The Issue (What was Missing)

The main issue addressed in this Pull Request was the absence of ready-to-use Bronze, Silver, and Snowflake templates within the bruin-data/bruin project. Bruin is built to help data teams structure ELT pipelines more efficiently, but without predefined templates, users had to manually create their own configurations every time they wanted to model data using the commonly accepted Medallion Architecture (Bronze → Silver → Gold) or integrate with Snowflake's data warehouse format.

This lack of built-in templates made it harder for new users to get started, slowed down development workflows, and resulted in inconsistencies across teams. Developers had to repeatedly write similar boilerplate configurations, which increased onboarding time and made best practices harder to enforce. The absence of standardized templates also meant that projects could differ widely in folder structure, naming, transformations, and SQL patterns, reducing the overall uniformity and maintainability of pipelines created with Bruin.

7.5.2 The Solution (What Was Added)

This Pull Request introduces complete, ready-to-use Bronze, Silver, and Snowflake templates to solve these onboarding and consistency problems. These templates provide clear starting points for building data pipelines following the Medallion Architecture and Snowflake's modeling conventions.

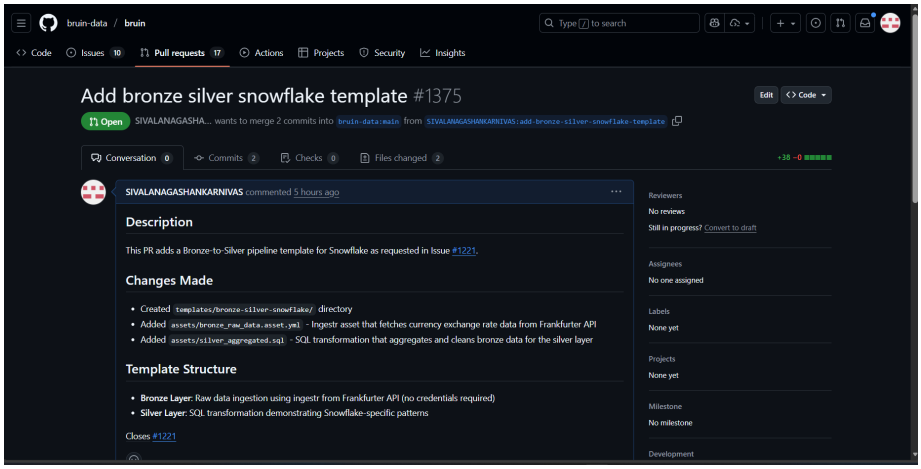
The update includes:

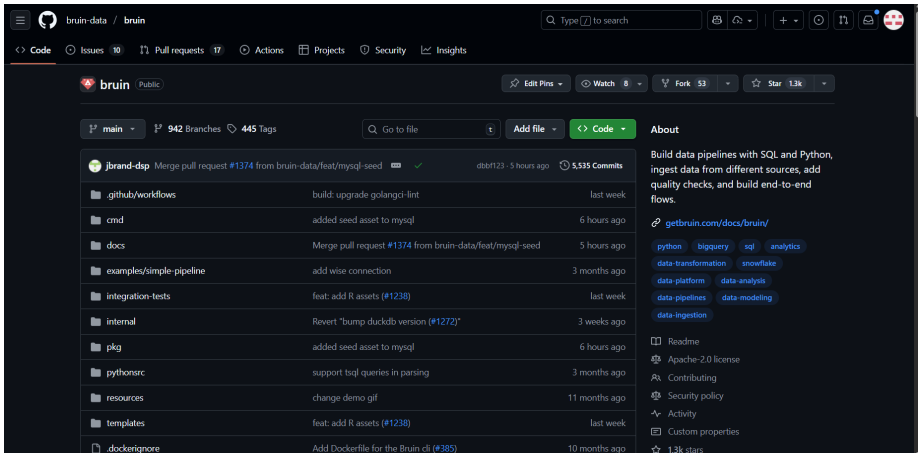
A Bronze template for raw ingestion, lightly processed data, and initial landing logic.

A Silver template for cleaned, standardized, and enriched data models.

A Snowflake-specific template that includes SQL structures, schema definitions, and warehouse-friendly transformation patterns tailored to Snowflake environments.

Each template is designed to follow Bruin's best practices, giving developers a reliable structure that they can reuse across projects. By offering consistent file layouts, recommended SQL patterns, and sample transformation logic, this PR significantly reduces setup time for new pipelines.





8 Linkedin Post Links

8.1 PR :

https://www.linkedin.com/posts/sivala-nivas-5a988b366_excited-to-share-my-first-open-source-contrib
utm_source=share&utm_medium=member_desktop&rcm=ACoAAFrmFZEBDXvqotgU7H6_F8uPpji0457RnF8

8.2 Journey Of Open Source :

https://www.linkedin.com/posts/sivala-nivas-5a988b366_how-i-started-with-linux-open-source-tools-ac
utm_source=share&utm_medium=member_desktop&rcm=ACoAAFrmFZEBDXvqotgU7H6_F8uPpji0457RnF8

8.3 Self Hosted Project :

https://www.linkedin.com/posts/sivala-nivas-5a988b366_we-did-it-our-team-successfully-self-hosted-a
utm_source=share&utm_medium=member_desktop&rcm=ACoAAFrmFZEBDXvqotgU7H6_F8uPpji0457RnF8