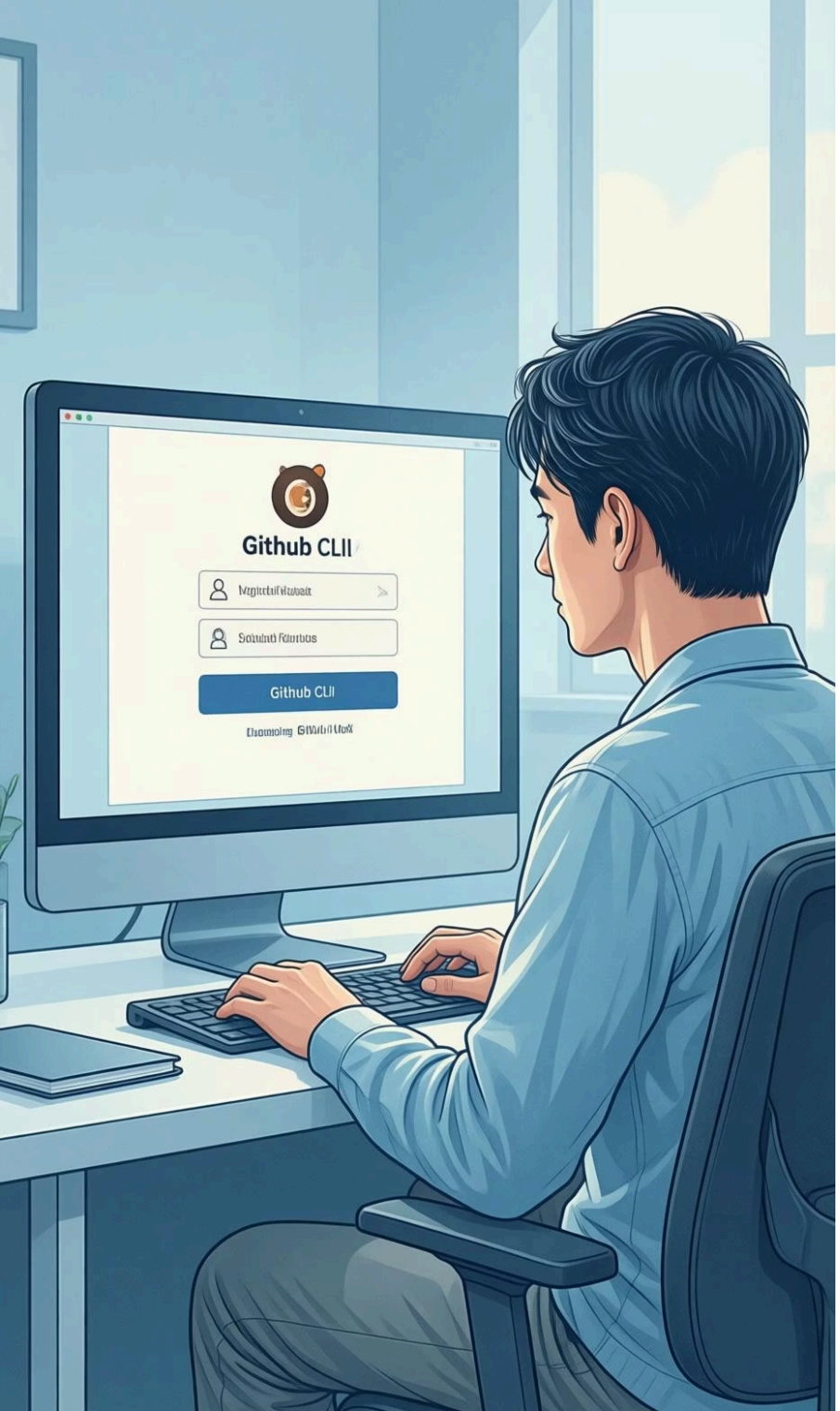
GitHub CLI & Pull Request Workflow for EcomiqX

This document outlines a clear and practical workflow for developers and team leads at EcomiqX, leveraging GitHub CLI for an efficient feature-branch pull request process.



1️⃣ Setting Up GitHub CLI

To begin, install the GitHub Command Line Interface on your preferred operating system. This powerful tool streamlines interactions with GitHub directly from your terminal.

1

Install CLI

- **Windows:** `winget install --id GitHub.cli`
- **Mac:** `brew install gh`
- **Linux:** `sudo apt install gh`

2

Login to GitHub

Execute `gh auth login` and follow the prompts. Select **GitHub.com**, **HTTPS**, and opt for **Browser login** for a seamless authentication experience.

2️⃣ Developer Workflow: Feature Branching

For feature development, always create and work on a dedicated feature branch. This isolates your changes and maintains the stability of the main development lines.

01

Create & Switch Branch

Begin by creating and switching to your new feature branch: `git checkout -b feature/login-api`.

02

Make Code Changes

Implement your new features or bug fixes. Write clean, well-documented code following EcomiqX standards.

03

Stage & Commit Changes

Stage your changes: `git add .`, then commit with a clear message: `git commit -m "Login API implemented"`.

Pushing & Creating a Pull Request

Once your feature branch is ready, push it to GitHub and create a Pull Request (PR) to initiate the review process.

- **Push Branch to GitHub** `git push origin feature/login-api` sends your local branch to the remote repository.
- **Create Pull Request** Use the GitHub CLI to create the PR: `gh pr create --base develop --head feature/login-api --title "Login API" --body "Implemented login REST API"`.
  - ✅ `--base develop`: This specifies the target branch for merging.
  - ✅ `--head feature/login-api`: This indicates your source branch with the changes.

3️⃣ Team Lead / Reviewer Workflow

Team leads and reviewers play a crucial role in maintaining code quality and project integrity. GitHub CLI simplifies the review and merge process.

List All PRs

`gh pr list` shows all open pull requests, helping you track pending reviews.

View PR Details & Diff

Examine a specific PR with `gh pr view` and review changes using `gh pr diff`.

Approve PR

Once satisfied, approve the PR: `gh pr review --approve`.

Merging and Branch Deletion

Merging a Pull Request correctly is vital for a clean repository history. Always use the GitHub CLI's merge functionality to ensure proper closure.

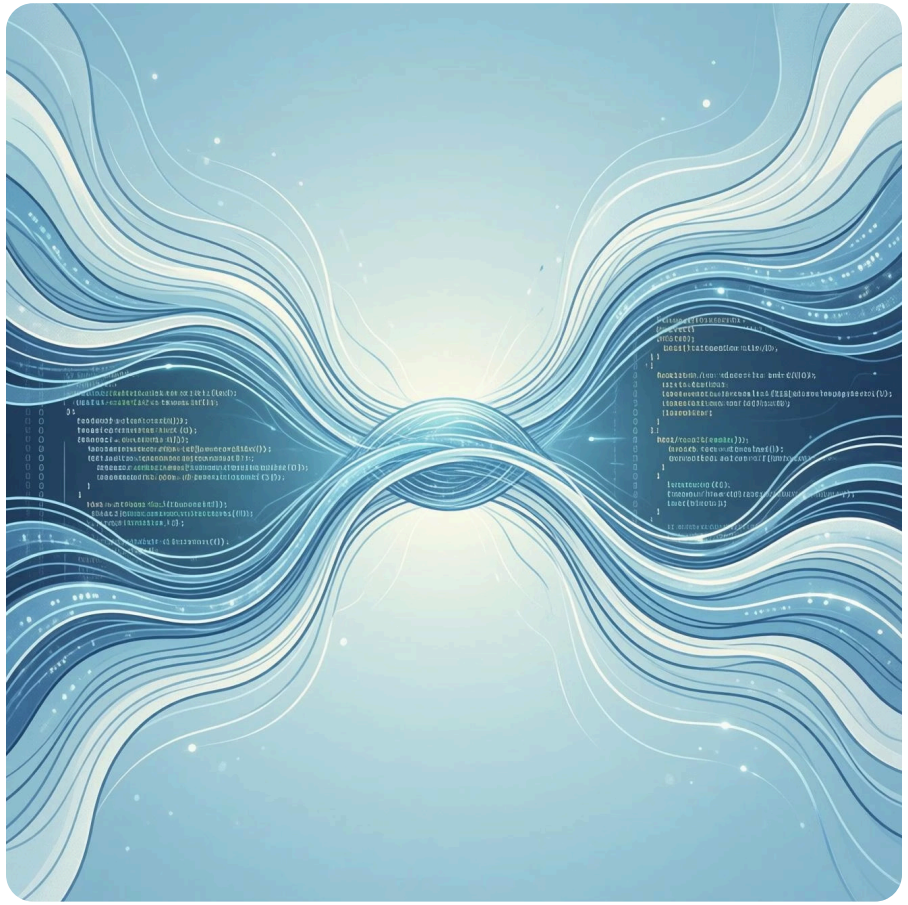
Merge PR

Use `gh pr merge --merge` to integrate changes into the base branch.

> **Optional Merge Strategies:**
>
> - `--squash`: Combines all commits into one for a cleaner history.
> - `--rebase`: Reapplies commits on top of the target branch history.

Delete Feature Branch

After a successful merge, automatically delete the feature branch to keep the repository clean: `gh pr merge --delete-branch`.



4️⃣ Alternative: Local Git Merge (Not Recommended)

While possible, performing a local Git merge is generally discouraged as it bypasses the comprehensive benefits of GitHub's Pull Request workflow.

Why avoid local merges? They often leave PRs open on GitHub, disrupting tracking and continuous integration/delivery processes.

- **Checkout & Pull Develop:** `git checkout develop`, then `git pull origin develop`.
- **Merge Feature Branch:** `git merge feature/login-api`.
- **Push to Develop:** `git push origin develop`.

> ⚠️ PR Stays Open: After a local merge, the PR on GitHub remains open and must be closed manually using `gh pr merge`.

5️⃣ Best Practices for EcomiqX

Adhering to these best practices ensures a smooth, collaborative, and maintainable development process for all EcomiqX projects.

Feature Branches

Always work on a feature branch; never directly on 'main' or 'develop'.

Commit Before Push

Ensure all local changes are committed before pushing to the remote.

Use `gh pr merge`

Leverage this command for closing PRs, not just 'git merge'.

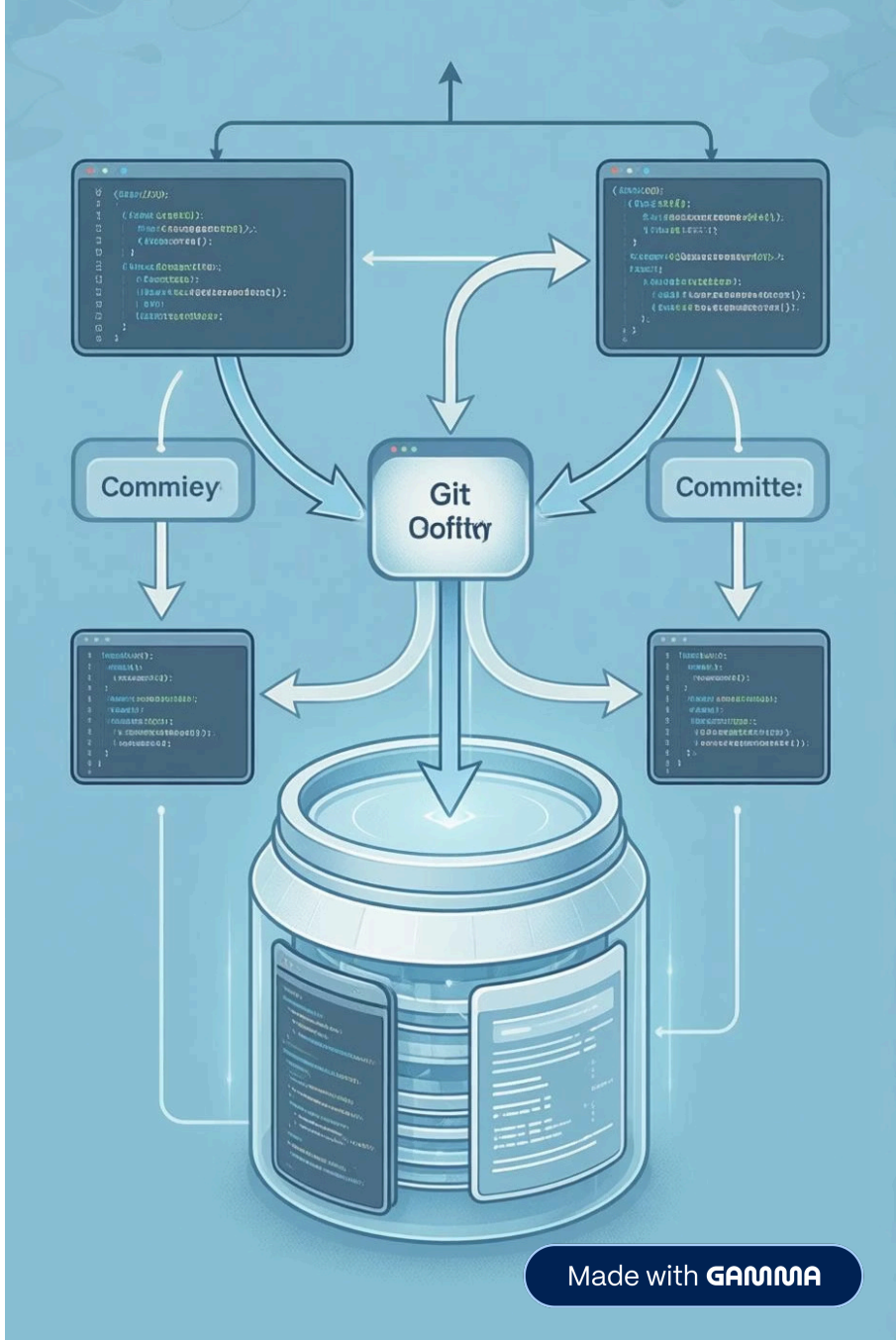
Delete Branches

Keep your repository clean by deleting branches after merging.

Adopting Git Flow

EcomiqX employs a simplified Git Flow model to structure its repository and manage releases effectively.

- **Feature branches** encapsulate new developments or fixes.
- **Develop branch** integrates completed features for testing.
- **Main branch** holds stable, production-ready code.



6️⃣ Optional: GitHub Copilot CLI Integration

Enhance your command-line experience with GitHub Copilot CLI, offering AI assistance for Git commands and explanations.

Install Copilot extension: `gh extension install github/gh-copilot`

> **Examples:**
>
> - `gh copilot suggest "create git branch for login feature"`
> - `gh copilot explain "git merge conflict"`

✅ Professional GitHub Workflow Summary for EcomiqX:

Developer: feature branch → commit → push → create PR

Reviewer / Lead: review → approve → merge → delete branch

This workflow ensures a clean history, tracked code reviews, intact CI/CD, and smooth team collaboration.