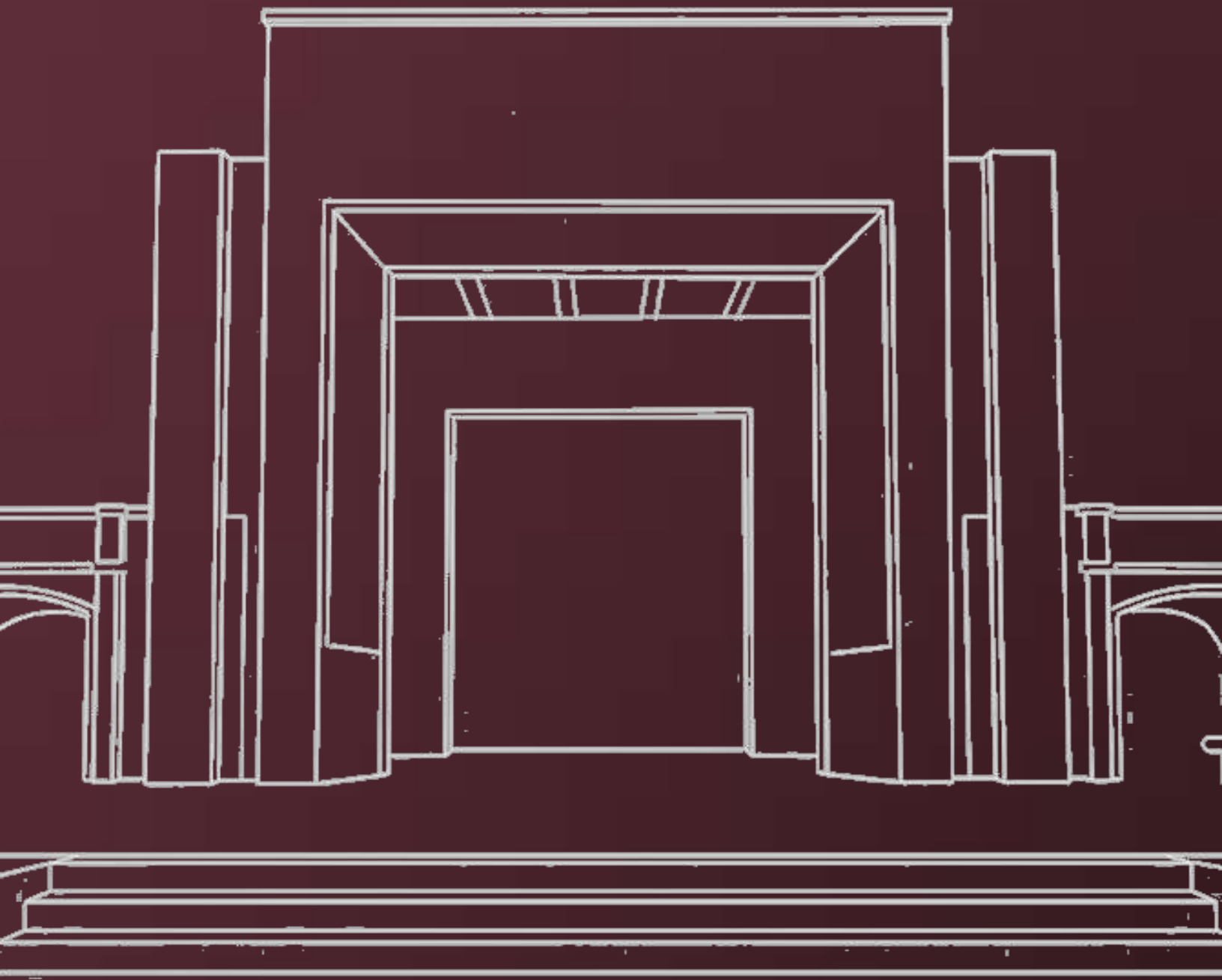
IEEE Robotics & Automation Society

IEEE RAS CUI · LHR

Open source Dev Day



Git & GitHub Fundamentals

Git is a distributed Version Control System (VCS) that tracks changes in source code locally on your machine, allowing you to revert to previous states and collaborate without overwriting work. GitHub is a cloud-based hosting platform for Git repositories that adds collaboration tools like Pull Requests, Issues, and Actions.

Feature	Git	GitHub
Type	Command-line software / Local tool	Cloud-based web service / Hosting platform
Storage	Local machine (.git folder)	Remote servers
Internet	Works completely offline	Requires internet to sync and collaborate
Core Role	Tracks code history and versions	Hosts code, manages teams, and reviews PRs

1. Initial Setup & Configuration

Before making your first commit, configure your identity so Git attaches your name and email to every snapshot you record.

Set your global username and email

git config --global user.name "Your Name"
git config --global user.email "you@example.com"

Set the default branch name to main
git config --global init.defaultBranch main

Verify your configuration
git config --list

2. Core Git Architecture (The Three States)

Understanding how files move through Git is critical to avoiding lost work:

1. Working Directory: Your active project folder where you create, edit, and delete files.
2. Staging Area (Index): A temporary holding zone where you prepare and review specific changes before saving them permanently.

3. Local Repository (.git directory): The database where Git permanently stores committed snapshots of your project history.

3. Essential Local Workflow Commands

Initialize a new local repository in the current folder

git init

Check the current state of files (untracked, modified, staged)

git status

Stage specific files or all changes for the next commit

git add filename.txt

or

git add .

Save staged changes permanently with a descriptive message

git commit -m "feat: add user login validation"

5. View the commit history

git log --oneline --graph –all

Ignoring Unwanted Files (.gitignore)

Create a file named .gitignore in your project root to prevent temporary files, dependencies, or secrets from being tracked.

Plaintext

Example .gitignore

node_modules/

.env

***.log**

build/

4. Branching & Merging

Branches allow you to isolate new features or bug fixes from your stable main codebase.

Bash

List all local branches (* indicates current branch)

git branch

Create new Branch

git branch newBranch

switch between branches

git switch newBranch

Merge a feature branch into your current branch (e.g., main)
git merge newBranch

Delete a branch after merging
git branch -d newBranch

Handling Merge Conflicts
A merge conflict occurs when two branches modify the exact same lines of a file. Git pauses the merge and marks the file:

5. Undoing Mistakes
git revert [COMMIT HASH]
Get commit hash with “git log” command

6. Connecting to GitHub (Remote Workflows)
git clone https://github.com/username/repo-name.git

Link an existing local repo to a new GitHub repo
git remote add origin https://github.com/username/repo-name.git

Verify linked remote URLs
git remote -v

Push local commits to GitHub (first time sets upstream tracking)
git push -u origin main

Subsequent pushes
git push

Fetch and merge the latest changes from GitHub into your local branch
git pull origin main

7. Open-Source & Team Collaboration Workflow
When contributing to projects you do not own (or working in teams), follow the Fork & Pull Request model:

1. Fork: Click Fork on the target GitHub repository to create a personal copy under your own GitHub account.

2. Clone: Clone *your* fork to your local machine (git clone <your-fork-url>).
3. Branch: Create a dedicated branch for your contribution (git switch -c fix/typo-in-readme).
4. Commit & Push: Make your changes, commit them, and push the branch to *your* fork (git push -u origin fix/typo-in-readme).
5. Pull Request (PR): Go to GitHub and click Compare & pull request. Write a clear title and summary explaining what your code changes and why.
6. Code Review & Merge: Project maintainers review your code, request changes if necessary, and merge your branch into the main project.

8. Best Practices Cheat Sheet

- Commit Early, Commit Often: Keep commits small and focused on one logical change.
- Write Clear Commit Messages: Use imperative verbs (e.g., "Add search bar" instead of "Added search bar").
- Never Push to main Directly: Always work on feature branches and merge via Pull Requests.
- Pull Before You Push: Always run git pull before starting new work or pushing to minimize merge conflicts.
- Never Commit Secrets: Keep API keys, passwords, and .env files out of version control.

WE THE RAS
IEEE RAS COMSATS Lahore