Computing at Scale

Lecture 3: Git, GitHub, and Code Review

Jacob Merson

January 13, 2024

Logistics

Other Logistics

- Homework 0 Questions (due 1/16)
- Office Hours. Fill out Survey on Zulip.

Code Reviews

What is a Code Review?

- Code reviews are a way to improve the quality of the code getting added to a codebase.
- They help catch bugs, improve style, and train junior developers.
- Often done in a pull request on GitHub, but they can be done in person.

What should you be looking at in a code review?

- Style
- Documentation
- Complexity and Structure
- Performance
- Testability
- Maintainability and Reusability
- Security (typically less important in HPC)

How to perform a code review?

Read the code

- Focus on the changes, not the entire codebase.
- Does the style match the rest of the codebase? naming conventions, indentation, etc. tools like clang-format can help.
- Is there documentation of the new code? Does it address what it does and how to use it?
- Are there any obvious bugs, or performance issues?
- Is the code easy to read and understand? Is it well organized? Are function names clear?
- Can raw loops be replaced with standard library algorithms?
- Note any questions or concerns.

How to perform a code review? (cont.)

Run the code and tests

- Does the code compile?
- Are there any compiler warnings?
- Are there any runtime errors?
- Does the code do what it is supposed to do?
- Are there test cases for all new functionality?
- Are there test cases that you think should be added? I.e., corner cases that are not covered.
- Do all of the tests pass?
- Are there any performance regressions?
- Does the code scale to the expected problem size?

How to perform a code review? (cont.)

Provide feedback

- Review the code with the author
- Discuss the changes and any concerns you have.
- Make sure the author understands your comments.
- Make sure you understand the author's reasoning.
- Make sure the author knows what changes need to be made.

How to perform a code review? (cont.)

Build Consensus and Approve the code

- Once you are satisfied with the changes, approve the code.
- If you are not satisfied, request changes.
- If you are unsure, ask for a second opinion.
- If you are not the right person to review the code, ask the right person to review it.
- Repeat the process until consensus is reached.

What should you *not* do in a code review?

- Be mean or condescending.
- ”My way or the highway” attitude. Code reviews are about building consensus.

All this seems like it requires a pretty sophisticated way of seeing what parts of the code have changed. How do we do that?

Git and GitHub

Git

- Git created by Linus Torvalds in 2005 for the Linux operating system.
- De-facto standard for version control in the software industry.
- Allows you to track changes to your codebase over time.
- Allows you to collaborate with others on the same codebase.
- Allows you to revert changes, and to see who made what changes.
- It can be used to manage multiple branches of the same codebase.
- Stores snapshots, not diffs.
- Distributed, meaning that operations are local.
- Hard to remove data from a git repository. (don't store secrets)

Basic Workflow

1. Modify files in your working tree.
2. Stage the changes you want to be into the next commit. Use `git add`.
3. Commit changes which stores snapshot of the changes. Use `git commit`.
4. Pull changes from the remote server. Use `git pull`.
5. Merge changes from the remote server. Use `git merge`.
6. Push changes to the remote server. Use `git push`.

`https://git-scm.com/book/en/v2/Getting-Started-What-is-Git%3F`

GitHub

- GitHub is a web-based platform for hosting git repositories.
- Provides a web interface for viewing code, issues, and pull requests.
- Provides tools for code review
- Provides tools for CI/CD
- Provides issue tracking and documentation tools

Basic Git Commands

- `git clone <url>` - Clone a repository from a remote server.
- `git add <file>` - Add a file to the staging area.
- `git commit -m "message"` - Commit the changes in the staging area.
- `git push` - Push the changes to the remote server.
- `git pull` - Pull the changes from the remote server.
- `git status` - Show the status of the working directory.
- `git log` - Show the commit history.

Initial Git Setup

- `git config --global user.name "Your Name"`
- `git config --global user.email youremail@email.com`
- `git config --global core.editor vim`
- `git config --global init.defaultBranch main`

Global settings are stored in `/.gitconfig` Project specific git settings are stored in `.git/config`

Demo

- Create new git repository. `git init`
- Add a README file to the repository with my name/info.
- Add Git ignore file.
- Commit the file.
- Branch.
- Create remote and push to github.
- Create fork and push to fork.
- Create pull request.
- pull request review.
- Show .git / config

Git Resources

- https://git-scm.com/book/en/v2
- https://docs.github.com/en/get-started

In Class Git Exercise

1. Create a github account. `https://docs.github.com/en/get-started/start-your-journey/creating-an-account-on-github`
2. Add your ssh key to your github account. `https://docs.github.com/en/authentication/connecting-to-github-with-ssh/adding-a-new-ssh-key-to-your-github-account`
3. Fork the course repository. `https://github.com/LACES-LAB/computing-at-scale-demo-2025`
4. Create a new markdown file with your name in lowercase separated by dashes as the title. Add your name, major, and a paragraph about your research to this file.
5. Commit your changes and push them to your fork.
6. Create a pull request with your changes.
7. Updated the README file to include a link to your file.

Resolving Merge Conflicts

- Merge conflicts occur when two branches have made changes to the same file.
- Git will not allow you to merge until the conflicts are resolved.
- To resolve a conflict, edit the file to remove the conflict markers.
- Add the file to the staging area.
- Commit the file.

Demo

- Status / log
- Stash
- Removing files from the staging area.
- Renaming / moving files.
- Deleting files.
- Reverting changes.
- Checking out old commits.
- Merging branches.
- Resolving merge conflicts.
- Clone

More Git Practice

Use the remainder of class to practice using git. Go through the examples here: `https://learngitbranching.js.org/?locale=en_US`.