Collaboration Tools in Software Engineering

Stepan Bolotnikov
Lecture 6: Advanced Git usage
By the end of this lecture, you will know:

- What is Continuous Integration and Continuous Deployment
- Why they are useful
- What are Git hooks
- What kind of hooks Git provides
- How to use Git hooks
After writing code

- Source code must be tested for errors continuously
- Developers should write automatic tests that can be run
- Instead of triggering tests automatically, would be better if system did it for us before the new code is committed

- Deploy application to test server after a newer code is committed
- Rebuild and deploy each time someone pushes a change
- Make the latest version available to customers and testers at all times
- Must be done automatically, otherwise waste of resources
CI and CD

● Continuous Integration
  ○ Testing all changes to codebase automatically and as early as possible
  ○ Every time a change is committed, a suite of test scripts is automatically run
  ○ Developers informed about errors

● Continuous Deployment
  ○ Automatically making the latest stable (tested) version of your application available at all times
  ○ When CI tests pass, a new version is built and deployed
How to CI/CD

● Tools provided by repository hosting
  ○ GitLab CI
  ○ BitBucket Pipelines

● Integrations
  ○ GitHub doesn’t have own CI/CD tools
  ○ Integrates other tools
  ○ Travis CI, Jenkins, AppVeyor and others
  ○ https://github.com/marketplace/category/continuous-integration

● Separate software

● Built-in Git tools - hooks
Git hooks

- Allow to run custom scripts when an event happens
- Available for client-side (local repositories) and server-side (remote repositories)
- Scripts can be written in any language
- Every time a repository is created, example scripts are added
Types of Git hooks

1. Commiting hooks
2. Other client-side hooks
3. Email workflow hooks
4. Receiving hooks
Commiting hooks

- Used on the client side during the committing process.
- Validate changeset
- Run tests
- Validate commit message

1. Pre-commit
2. Prepare-commit-msg
3. Commit-msg
4. Post-commit
Pre-commit hook

- Fired after `git commit` command, before commit message
- Mostly for validating changeset
- Run tests
- Check for documentation, whitespace, etc
- Can abort commit if anything is wrong
Prepare-commit-msg hook

- Fired after commit message generated, before editor is opened
- Mostly used to generate default commit message
- When merging, squashing or amending
Commit-msg hook

- Fired after entering commit message
- Can abort commit
- Used to validate commit message
Post-commit hook

- Fired after the commit is created successfully
- Notify an external system
- Send e-mails to collaborators
Other client-side hooks

1. Pre-rebase
2. Post-rewrite
3. Post-checkout
4. Post-merge
5. Pre-push
6. Pre-auto-gc
Pre-rebase hook

- Fired before rebase is applied
- Can abort rebase
- Prevent rebasing commits that were already pushed
- Prevent rebasing commits that are already on this branch
- Backup data that is being rebased
Post-rewrite hook

- Triggered by commands that rewrite commits
  - git rebase
  - git commit --amend
- Set up working area
- Automatically generate files that is not being kept in the repository
Post-checkout hook

- Fired after a successful `git checkout`
- Set up working area
- Automatically generate files that is not being kept in the repository
Post-merge hook

- Fired after a successful `git merge`
- Set up working area
- Automatically generate files that is not being kept in the repository
Pre-push hook

- Triggered during `git push`
- After data from remote is updated
- Before changes are transferred to the remote
- Can abort push
- Validate changes, commit messages etc
Pre-auto-gc hook

- `git gc` command runs Git’s build-in garbage collection
- Gets rid of unnecessary files
- Optimises local repository
- `git gc` is periodically ran automatically

- Hook fired before garbage collection happens
- Can abort garbage collection
- Notify user that garbage collection is happening
- Allow user to postpone/abort garbage collection
Email workflow

- Nowadays open source projects organised on services like GitHub, GitLab, BitBucket etc
- Collaboration done with forking, pull requests
- Alternatively Git allows workflow for collaborating over e-mail
Email workflow commands