Collaboration Tools in Software Engineering

Stepan Bolotnikov
Lecture 6: Git hooks and CI
Today

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
- How to set up CI in Github
After writing code

- Source code must be tested for errors continuously
- Developers should write tests that can be run
- Instead of triggering tests manually, would be better if system did it for us when code changes

- Deploy application to test server after a newer code is committed
- Rebuild and deploy each time the application is changed
- 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
CI service

- Allows to define scripts that are run automatically
- Visualises current state of the application
- Allows monitoring build
- Provides API to hook into other services
  - Emails, notifications, etc
How to CI/CD

- Tools provided by repository hosting
  - GitLab CI
  - Bitbucket Pipelines
  - Github Actions

- Integrations
  - Github integrates with many other cloud-based tools
  - Travis CI, AppVeyor and others
  - [https://github.com/marketplace/category/continuous-integration](https://github.com/marketplace/category/continuous-integration)

- Separate software
  - Jenkins, Teamcity

- 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 (but they’re disabled)
Types of Git hooks

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

- Used on the client side during the commiting 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
- Automatically style code
  - Prettier for JavaScript
- 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
- Customise message for merging, squashing or amending commits
Commit-msg hook

- Fired after entering commit message
- Can abort commit
- Used to validate commit message
- Force developers to use specific form of commit messages
Post-commit hook

- Fired after the commit is created successfully
- Cannot abort commit
- Notify an external system
- Send emails to collaborators
Receiving hooks

- Run on the server side (in a \texttt{--BARE} repository)
- Triggered during and after someone does \texttt{git push} to the repository
- Can be used to enforce a policy that collaborators can’t ignore
- Can be used to output information to the pusher

1. Pre-receive
2. Update
3. Post-receive
Pre-receive hook

- Triggered when someone does a `git push`
- Triggered once for each push
- Can abort push
- Inspect & validate incoming changes
Update hook

- Triggered when someone does a `git push`
- Triggered once for every branch that is being pushed
- Can abort push of this particular branch
- The other branches can still succeed
- Inspect and validate the changes being pushed to this branch
Post-receive hook

- Triggered when someone does a `git push`
- Triggered after pushing process completes successfully
- Cannot abort push
- Notify users
- Send emails
- Update other services
- Run continuous integration
- Run continuous deployment
- Update issue tracker
### Other client-side hooks

<table>
<thead>
<tr>
<th>Hook Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Pre-rebase</strong></td>
<td></td>
</tr>
</tbody>
</table>
  - Fired before rebase is applied  
  - Can abort rebase  
  - Prevent rebasing existing commits |
| **Post-rewrite** |  
  - Fired by commands that rewrite commits  
  - Can not abort  
  - Set up working area |
| **Post-checkout** |  
  - Fired after successful git checkout  
  - Set up working area |
| **Post-merge** |  
  - Fired after successful git merge  
  - Set up working area |
| **Pre-push** |  
  - Fired during git push  
  - After data from remote is updated  
  - Before changes are pushed to the remote  
  - Can abort push  
  - Validate changes, commit message, etc |
| **Pre-auto-gc** |  
  - Fired by git gc  
  - Git gc periodically run automatically  
  - Garbage collection  
  - Can abort garbage collection |
## Summary

<table>
<thead>
<tr>
<th>Committing hooks</th>
<th>Receiving hooks</th>
<th>Other client/side hooks</th>
<th>Email workflow hooks</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Pre-commit</td>
<td>1. Pre-receive</td>
<td>1. Pre-rebase</td>
<td>1. Applypatch-msg</td>
</tr>
<tr>
<td>4. Post-commit</td>
<td></td>
<td>4. Post-merge</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>5. Pre-push</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>6. Pre-auto-gc</td>
<td></td>
</tr>
</tbody>
</table>

- **Committing hooks**: These hooks are executed during the commit process, allowing developers to perform actions before and after commit messages are written to the repository.
- **Receiving hooks**: These hooks are executed when a push or receive operation occurs, allowing developers to perform actions related to the incoming changes.
- **Other client/side hooks**: These hooks are executed when a rebase, rewrite, or checkout operation occurs, allowing developers to perform actions related to the local copy of the repository.
- **Email workflow hooks**: These hooks are executed when patches are applied via email, allowing developers to perform actions related to the email-based workflow.
What is a Git hook technically?

- A program
- Written in any language
- Runnable from command line
- Specified name of file
  - No extension
- In specified location
- Git provides example hooks in every new repository
Where hooks are

$ git init
Initialized empty Git repository in .git/
$ cd .git/
$ ls -la

total 0

```
drwxrwxrwx 1 stepan stepan 4096 May 4 19:06 .
drwxrwxrwx 1 stepan stepan 4096 May 4 19:06..
drwxrwxrwx 1 stepan stepan 4096 May 4 19:06 branches
-rwxrwxrwx 1 stepan stepan  93 May 4 19:06 config
-rwxrwxrwx 1 stepan stepan  73 May 4 19:06 description
-rwxrwxrwx 1 stepan stepan  23 May 4 19:06 HEAD
-drwxrwxrwx 1 stepan stepan 4096 May 4 19:06 hooks
-drwxrwxrwx 1 stepan stepan 4096 May 4 19:06 info
-drwxrwxrwx 1 stepan stepan 4096 May 4 19:06 objects
-drwxrwxrwx 1 stepan stepan 4096 May 4 19:06 refs
```

Where hooks are

$ git init
Initialized empty Git repository in .git/
$ cd .git/
$ ls -la

Where hooks are
Example hooks

$ cd hooks
$ ls -la

```
total 28
-rwxrwxrwx 1 stepan stepan  478 May  4 19:06 applypatch-msg.sample
-rwxrwxrwx 1 stepan stepan  896 May  4 19:06 commit-msg.sample
-rwxrwxrwx 1 stepan stepan  189 May  4 19:06 post-update.sample
-rwxrwxrwx 1 stepan stepan  424 May  4 19:06 pre-applypatch.sample
-rwxrwxrwx 1 stepan stepan 1642 May  4 19:06 pre-commit.sample
-rwxrwxrwx 1 stepan stepan 1239 May  4 19:06 prepare-commit-msg.sample
-rwxrwxrwx 1 stepan stepan 1348 May  4 19:06 pre-push.sample
-rwxrwxrwx 1 stepan stepan 4898 May  4 19:06 pre-rebase.sample
-rwxrwxrwx 1 stepan stepan 3610 May  4 19:06 update.sample
```
Adding a hook

- Rename one of the default hooks, removing “.sample”
- Write your own script
  - Name it as the hook that you want to run it
  - Place it in .git/hooks/
Example pre-commit hook

echo "No committing allowed"
exit 1

- Outputs a message
- Aborts the commit
- Put the script into .git/hooks/pre-commit
Example hook being triggered

$ git commit
No committing allowed
$
Non-blocking pre-commit hook

echo "No committing allowed"
#exit 1
Non-blocking pre-commit hook

echo "No committing allowed"
#exit 1

- Commented out.
- Default behaviour is “exit 0”, which doesn’t abort commit. Same goes for all other blockable hooks.
Non-blocking hook being triggered

$ git commit -m "Test commit"
No committing allowed
[master 3ef2e34] Test commit
$
Non-blocking hook being triggered

$ git commit -m "Test commit"
No committing allowed
[master 3ef2e34] Test commit
$

Message still output

Commit not blocked
CI and CD with Git hooks

- **Client-side**
  - Pre-commit and/or pre-push hook to run tests

- **Server-side**
  - Update hook to verify
  - Post-receive hook to run tests
  - Post-receive hook to deploy
CI & CD in Github

- Git hooks only supported in Github Enterprise
- Possible to set up CI/CD using Github Actions
- Supports “webhooks”
  - Send HTTP requests when certain events happen
- Allows other services to integrate
- Services can read and write repository data
  - Code quality
  - Code review
  - CI & CD
  - Monitoring
  - Project management
  - etc
Github Actions

- Docker-based automation workflow
- Can define containers to run tasks
- Can be triggered on any Github event
- Can use containers, scripts, and workflows created by other people
- [https://github.com/marketplace?type=actions](https://github.com/marketplace?type=actions)
Webhooks in Github

- Build your own web application
- Some existing services support it
- Receive a HTTP POST request when event happens
- Repository events
- Wiki, Issues, Pull requests, Project boards etc
Where to find webhooks

Webhooks

Webhooks allow external services to be notified when certain events happen. When the specified events happen, we’ll send a POST request to each of the URLs you provide. Learn more in our Webhooks Guide.
Webhooks

Webhooks allow external services to be notified when certain events happen. When the specified events happen, we'll send a POST request to each of the URLs you provide. Learn more in our Webhooks Guide.
Adding a webhook

We'll send a POST request to the URL below with details of any subscribed events. You can also specify which data format you'd like to receive (JSON, x-www-form-urlencoded, etc). More information can be found in our developer documentation.

Payload URL *
https://example.com/postreceive

Content type
application/x-www-form-urlencoded

Secret

Which events would you like to trigger this webhook?
- Just the push event.
- Send me everything.
- Let me select individual events.

Active
We will deliver event details when this hook is triggered.

Add webhook
Populated webhooks list

Webhooks

Webhooks allow external services to be notified when certain events happen. When the specified events happen, we’ll send a POST request to each of the URLs you provide. Learn more in our Webhooks Guide.

- [✓] https://example.com/ (fork, pull_request, push, an...)
- [✗] https://example.ee/ (gollum, issues, project, proj...)

Add webhook
Edit | Delete
Where to find integrations

- [https://github.com/marketplace](https://github.com/marketplace)
- Includes free and non-free applications and services
- Some have free trial periods
Github Marketplace

GitHub Marketplace
Tools to build on and improve your workflow

- **ZenHub**
  Agile Task Boards, Epics, Estimates & Reports, all within GitHub's UI

- **AccessLint**
  Find accessibility issues in your pull requests

- **Waffle**
  Automated project management powered by your GitHub issues & pull requests

- **POEditor**
  Manage your software localization collaboratively

---

**Apps with free trials**
Discover new ways to build better—now with a free 14-day trial.

- **Dependabot**
  Automated dependency updates for Ruby, Javascript, Python, PHP, Elixir, Rust and Java

- **lsdiff**
  Agile project management inside GitHub, without access to code

- **Crowdin**
  Agile localization for your projects

- **ZenHub**
  Agile Task Boards, Epics, Estimates and Reports, all within GitHub's UI

- **Cadecov**
  Groups, merge, archive and compare coverage reports

- **Travis CI**
  Test and deploy with confidence
CI tools in Github Marketplace

Automatically build and test your code as you push it to GitHub, preventing bugs from being deployed to production.

- **Cloud 66 for Rails**: Build, deploy, and maintain your Rails apps on any cloud or server.
- **Buddy**: One-click delivery automation for Web Developers.
- **Percy**: Continuous visual testing and reviews for web apps.
- **AppVeyor**: Cloud service for building, testing and deploying Windows apps.
- **Semaphore**: Test and deploy at the push of a button.
- **Travis CI**: Test and deploy with confidence.
- **CircleCI**: Automatically build, test, and deploy your project in minutes.
- **Cloud 66 Skypcap**: Skypcap is a container native CI/CD tool.

Now offering free trials
Free for 14 days

View apps
Jenkins

- Stand-alone application
- Free and open-source
- Have to install and set up by yourself
- Can be integrated with a Github repository
Nevercode

- Cloud-based CI service for mobile applications
- Developed in Tartu
- Several pricing schemes
- Free trial
- Can be integrated with Github
Travis CI

- Cloud-based CI platform
- Integrates directly into Github
- Several pricing schemes
  - Free trial
  - Different number of scripts run simultaneously
  - $69-$349/month
- Free for open-source projects
Setting up Travis

- Sign up with your Github account
- Enable for the repository
- Add a configuration file to the repository
Summary

Today we covered

- What is CI & CD
- What are Git hooks
- How to add Git hooks
- How to set up CI in Github
Practical session

● Registering for Travis CI
● Setting up testing with Travis CI
● Creating Git hooks
References

- https://css-tricks.com/continuous-integration-continuous-deployment/
- https://github.com/marketplace