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 someone pushes a change
- Make the latest version available to customers and testers at all times
- Must be done automatically, otherwise waste of resources
Cl 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

- 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](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. Committing hooks
2. Receiving hooks
3. Other client-side hooks
4. Email workflow hooks
Comitting 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
- 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 --BARE repository)
- Triggered during and after someone does `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>Pre-rebase</th>
<th>Post-rewrite</th>
<th>Post-checkout</th>
</tr>
</thead>
<tbody>
<tr>
<td>● Fired before rebase is applied</td>
<td>● Fired by commands that rewrite commits</td>
<td>● Fired after successful git checkout</td>
</tr>
<tr>
<td>● Can abort rebase</td>
<td>● Can not abort</td>
<td>● Set up working area</td>
</tr>
<tr>
<td>● Prevent rebasing existing commits</td>
<td>● Set up working area</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Post-merge</th>
<th>Pre-push</th>
<th>Pre-auto-gc</th>
</tr>
</thead>
<tbody>
<tr>
<td>● Fired after successful git merge</td>
<td>● Fired during git push</td>
<td>● Fired by git gc</td>
</tr>
<tr>
<td></td>
<td>● Set up working area</td>
<td>● Git gc periodically run</td>
</tr>
<tr>
<td></td>
<td></td>
<td>automatically</td>
</tr>
<tr>
<td></td>
<td></td>
<td>● Garbage collection</td>
</tr>
<tr>
<td></td>
<td></td>
<td>● Can abort garbage collection</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
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>5. Pre-push</td>
<td></td>
<td>5. Pre-push</td>
<td></td>
</tr>
<tr>
<td>6. Pre-auto-gc</td>
<td></td>
<td>6. Pre-auto-gc</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
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

```
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
```

Example hooks

$ cd hooks
$ ls -la

```
total 28
drwxrwxrwx 1 stepan stepan 4096 May  4 19:06 .
drwxrwxrwx 1 stepan stepan 4096 May  4 19:06 ..
-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 "Thou shalt not commit"
exit 1

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

$ git commit
Thou shalt not commit
$
$
Non-blocking pre-commit hook

echo “Thou shalt not commit”
#exit 1
Non-blocking pre-commit hook

echo "Thou shalt not commit"
#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"
Thou shalt not commit
[master 3ef2e34] Test commit
$

Non-blocking hook being triggered

$ git commit -m "Test commit"
Thou shalt not commit
[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

- Unlike many other services, GitHub doesn’t have its own CI/CD solution QUITE YET
- Cannot set receiving hooks
- 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
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.
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.
Adding a webhook

Webhooks / Add 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
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...)
  - **Edit** | **Delete**

- **https://example.ee/** (gollum, issues, project, proj...)
  - **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

- **issu.sh**
  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

- **Cadacov**
  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.
- **Semaphore**: Test and deploy at the push of a button.
- **Buddy**: One-click delivery automation for Web Developers.
- **Travis CI**: Test and deploy with confidence.
- **Percy**: Continuous visual testing and reviews for web apps.
- **CircleCI**: Automatically build, test, and deploy your project in minutes.
- **AppVeyor**: Cloud service for building, testing and deploying Windows apps.
- **Cloud 66 Skycap**: Skycap is a container native CI/CD tool.
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
NEW! GitHub Actions

Focus on what matters: code

With GitHub Actions you can automate your workflow from idea to production.

Sign up for the beta

Build, Test, and Deploy on push

Build
actions/docker
uses actions/docker
runs docker build -t octoverse

Powerful workflows to supercharge your repos

Workflows can be triggered by GitHub platform events (i.e. push, issue, release) and can run a sequence of serial or parallel actions in response.

Get octocat
.fetch-from-octodex/

Provision Database
hashicorp/terraform
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