Collaboration Tools in Software Engineering

Stepan Bolotnikov
Lecture 3: Working with repository history
Last time

- Configuration
- Repository creation
- Staging
- Resetting
- Committing
- Pull & Push
Today

At the end of today’s lesson, you will know:

● How Git manages commits
● How to view commit history
● How to undo changes
● How to change history
First look at commit history

$ git log
commit 7a6524a69d93a6ea3dbe8bdb737e4411f119c878
Author: Stepan Bolotnikov <stjoopa@gmail.com>
Date:   Wed Feb 28 18:31:22 2018 +0200

    Self-service mobile css

commit f754ce3d9abbea2f1bd187d7d019aed842a80d5b
Author: Stepan Bolotnikov <stjoopa@gmail.com>
Date:   Wed Feb 28 16:59:53 2018 +0200

    added the bottom info blocks
Self-service mobile css
SHA-1

- Secure Hash Algorithm 1
- Published in 1995 by US National Security Agency
- Cryptographic hash function
- Produces 160-bit hash value ("message digest")
- Commonly represented as 40-digit hexadecimal number
- Not considered secure against well-funded opponents since 2005
- In Git, used for data integrity, not security
Data integrity

- Each commit identified by hash
- Hash depends on file system snapshot; date; author; comment; previous commit
- Therefore hash depends on whole history
- Even a small change completely changes the hash
- Cannot change or remove a commit unnoticed

SHA1("The quick brown fox jumps over the lazy dog")
gives hexadecimal: 2fd4e1c67a2d28fced849ee1bb76e7391b93eb12

SHA1("The quick brown fox jumps over the lazy cog")
gives hexadecimal: de9f2c7fd25e1b3afad3e85a0bd17d9b100db4b3
Short SHA-1

7a6524a69d93a6ea3dbe8bdb737e4411f119c878

- Git can mostly figure out a commit by partial SHA-1 hash
- 4+ digits; unambiguous

7a6524a69d93a6ea3dbe

7a6524a69d

7a65
Git-log tool

$ git log <options> <path>

- Lists commits made in the reverse chronological order
- Many options to customise output
Limiting commits

- By default, git paginates commits and shows one page at a time
- Reverse chronological order
- Until the creation of the repository
Show n last commits

$ git log -5

- Shows 5 last commits
- \(-n\) accepts any integer
Skip n commits

$ git log --skip=5

- Skips the given number of commits before starting to show output

$ git log --skip=5 -10

- Skip 5, show 10
Limit commits by time

$ git log --since=2015-12-30 --until=2.weeks.ago

● --since shows start date
● --until shows end date
● Git accepts dates in several absolute and relative formats

Mon, 3 Jul 2006 17:18:43 +0200
2006-07-03 17:18:43 +0200
Mon Jul 3 15:18:43 2006
2006-07-03
2.years.3.months.ago
6am.yesterday
Limit commits by author

$ git log --author=Stepan

- Argument is assumed to be a regular expression
- Git separates author and committer
  - Author is the one who wrote the code
  - Committer is the one who added the commit
  - Author and committer can differ in several situations
    - Merging pull requests. Author: code author; committer: repository owner/merger
    - History rewrite/rebase. Author: code author; committer: rebaser

$ git log --author="Stepan Bolotnikov <stjoopa@gmail.com>"
Limit commits by commit message

$ git log --grep=<message>

- Argument is assumed to be a regular expression

$ git log --grep=[Aa]dded
commit 8158551085063c0380a957e4d52125204f7f5a27
Author: Stepan Bolotnikov <stjoopa@gmail.com>
Date: Wed Feb 28 16:31:39 2018 +0200
    Added loan list block
commit 82c700dab52958c79a704bd0d1f5575bb52f1abd
Author: Stepan Bolotnikov <stjoopa@gmail.com>
Date: Wed Feb 28 16:59:53 2018 +0200
    added the bottom info blocks
commit 40c23801d04987c4e8db1f93c45236b0ca6a4359
Author: Stepan Bolotnikov <stjoopa@gmail.com>
Date: Wed Feb 28 16:31:39 2018 +0200
    Added loan list block
Limit commits by content

$ git log -S <string>
$ git log -G <regex>

- -S finds commits that change the number of occurrences of given string
- -G finds commits that have added/removed lines that match the given regex
Difference between -S and -G

+    return !regexec(regexp, two->ptr, 1, &regmatch, 0);
...
-    hit = !regexec(regexp, mf2.ptr, 1, &regmatch, 0);

● Number of occurrences of “regexec(regexp” stayed the same

$ git log -G "regexec(regexp"
Finds this commit

$ git log -S "regexec(regexp"
Doesn’t find this commit
Limit commits by path

$ git log <path/to/file>

- Finds only commits that change given path
Many other options

https://git-scm.com/docs/git-log
Formatting output

- By default: SHA-1, author, date, message

$ git log
commit 7a6524a69d93a6ea3dbe8bdb737e4411f119c878
Author: Stepan Bolotnikov <stjoopa@gmail.com>
Date: Wed Feb 28 18:31:22 2018 +0200

    Self-service mobile css
Showing diff for each commit

$ git log --patch
commit 7a6524a69d93a6ea3dbe8bdbc737e4411f119c878
Author: Stepan Bolotnikov <stjoopa@gmail.com>
Date:   Wed Feb 28 18:31:22 2018 +0200

Self-service mobile css

diff --git a/scss/components/button-big.scss b/scss/components/button-big.scss
index 738de1e..95438d3 100644
--- a/scss/components/button-big.scss
+++ b/scss/components/button-big.scss
@@ -11,4 +11,11 @@ a.button-big {
       vertical-align: 2px;
       line-height: 1;
   }
diff --git a/scss/components/button-big.scss b/scss/components/button-big.scss
index 738de1e..95438d3 100644
--- a/scss/components/button-big.scss
+++ b/scss/components/button-big.scss
@@ -11,4 +11,11 @@ a.button-big {
 vertically-align: 2px;
 line-height: 1;
 }
+}^M
+@media screen and (max-width: $mobile_breakpoint) {
+ .button-big,^M
+ a.button-big {^M
+ padding: 10px 16px;^M
+ font-size: 24px;^M
+ }
+ }^M
+}
\ No newline at end of file
Showing change summary

$ git log --stat
commit 7a6524a69d93a6ea3dbe8bdb737e4411f119c878
Author: Stepan Bolotnikov <stjoopa@gmail.com>
Date:   Wed Feb 28 18:31:22 2018 +0200

    Self-service mobile css

    scss/components/button-big.scss            |  7 +++++++
    scss/components/info-tabs.scss             | 25 +++++++++++++++++++++++++++++++++++++
    scss/components/loan-list-block.scss       | 19 +++++++++++++++++++++
    scss/components/loan-table.scss            | 17 +++++++++++++++++++
    scss/components/button-block.scss          | 28 ++++++++++++++++++++++++++++++++++++--
    scss/components/well.scss                  | 35 ++++++++++++++++++++++++++++++++++++++
    selfservice.html                          | 18 ++++++++++++++++++-

7 files changed, 142 insertions(+), 7 deletions(-)
The most minimal summary

$ git log --oneline
7a6524a Self-service mobile css
f754ce3 added the bottom info blocks
2dab5cd Fixed typo in class name
More control

$ git log --pretty=<format>

- Pretty accepts one of pre-defined formats or a format string
- Pre-defined formats include oneline, short, medium, full, fuller, email, raw

$ git log --pretty=fuller
Viewing information about one commit

$ git show <hash>

- Shows information about one commit:
  - Author
  - Date
  - Message
  - Patch diff
External software for browsing repository history

- Git installation comes with the Gitk software for browsing repositories
- Git hosting services typically allow for viewing of history
- A great number of external software available
  - Sourcetree
- IDEs may have their own modules
  - IntelliJ IDEA
Gitk
Github commit history

No description, website, or topics provided.

Add topics

6 commits

1 branch

0 releases

1 contributor

Latest commit 0169cc0 11 days ago
## Github commit history

### Commits on Mar 9, 2018

- **Update asdasdasdfile**
  - Stopa committed 11 days ago

### Commits on Mar 8, 2018

- **Create asdasdasdfile**
  - Stopa committed 12 days ago

- **Merge branch 'master' of github.com:Stopa/cse-prac-1**
  - Stopa committed 12 days ago

### Commits on Mar 7, 2018

- **ascd**
  - Stopa committed 12 days ago

### Commits on Feb 23, 2018

- **My first commit**
  - Stopa committed 25 days ago

- **Initial commit**
  - Stopa committed 25 days ago
My first commit

Stopa committed 25 days ago

1 parent 0e37c6

commit d861bdf06e8bd3c1c1b2c5831748266538e8daeb8e

Showing 1 changed file with 2 additions and 1 deletion.

3 | README.md
...
1 | -# cse-prac-1
1 | +# cse-prac-1
2 | Stepan Bolotnikov

0 comments on commit d861bdf

Leave a comment

Attach files by dragging & dropping, selecting them, or pasting from the clipboard.

Styling with Markdown is supported

Comment on this commit
IntelliJ IDEA

https://www.jetbrains.com/help/idea/investigate-changes.html
Reverting a file

$ git checkout <revision> <file>

- Retrieves the file at a given version to the working copy and the index
- Can be done by hunks with --patch argument
Restoring a deleted file

- Find the last commit where file was present
- Check out file from that commit

OR

- Find commit where file was removed
- Check out file from <commit>~
- ~ at the end of SHA-1 hash means “parent”. Can be followed by number - “This many generations up”
Restoring a deleted file

$ git log <path>
fatal: ambiguous argument '<path>': unknown revision or path not in the working tree.
Use '--' to separate paths from revisions, like this:
'git <command> [<revision>...] -- [<file>...]'

- Because the file no longer exists, Git will not be able to understand if you are talking about a path, revision, branch or something else
- Use "--" to separate path from all other arguments

$ git log -- <path>
How to revert a file to what it was BEFORE a commit

$ git checkout <hash>~ <path>

- Checks out the commit before the given hash. Same as:
  
  $ git checkout <hash>~1

  $ git checkout <hash>~2

- Two commits before the given hash
- You will often see “HEAD~”. That means “commit that was before the current HEAD”
Rewriting history

- Recommended on local commits
- Therefore: use `git push` when you are sure you’re pleased with your code
- Treat it as “make public” and treat pushed code as final

- As a rule of thumb, frequency of git commands in descending order:
  - Status
  - Add
  - Commit
  - Push
Changing the last commit

- Avoid “oops forgot this one file” commit messages
- Easiest commit history changing operation
- Can change any aspect of last commit
Amending a commit

1. Changing a commit message:

$ git commit --amend

- Opens editor with last message to allow changing
- Changes commit SHA-1
Amending a commit

2. Changing content of the commit:

$ git add <path>
$ git commit --amend

- Make necessary changes
- Stash changes
- Add to last commit
- Opens editor to edit message
- Changes SHA-1
Changing deeper commits

- There is no tool specifically for changing history
- Git rebase tool is typically used
- Formally “Reapply commits on top of another base tip” (e.g. from another branch)
- To change history: “reapply commits on top of the same base tip”
Git-rebase example

- Scenario: want to change something in three last commits

$ git rebase -i HEAD~3

- “Start an interactive (-i) rebase from the commit 3 commits (~3) before the last committed state (HEAD)”
Rebase script

pick 2dab5cd Fixed typo in class name
pick f754ce3 added the bottom info blocks
pick 7a6524a Self-service mobile css

# Rebase 1278e6a..7a6524a onto 1278e6a (3 command(s))
#
# Commands:
# p, pick = use commit
# r, reword = use commit, but edit the commit message
# e, edit = use commit, but stop for amending
# s, squash = use commit, but meld into previous commit
# f, fixup = like "squash", but discard this commit's log message
# x, exec = run command (the rest of the line) using shell
# d, drop = remove commit
#
# These lines can be re-ordered; they are executed from top to bottom.
#
# If you remove a line here THAT COMMIT WILL BE LOST.
#
# However, if you remove everything, the rebase will be aborted.
pick 2dab5cd Fixed typo in class name
pick f754ce3 added the bottom info blocks
pick 7a6524a Self-service mobile css

# Rebase 1278e6a..7a6524a onto 1278e6a (3 command(s))
#
# Commands:
# p, pick = use commit
# r, reword = use commit, but edit the commit message
# e, edit = use commit, but stop for amending
# s, squash = use commit, but meld into previous commit
# f, fixup = like "squash", but discard this commit's log message
# x, exec = run command (the rest of the line) using shell
# d, drop = remove commit
#
# These lines can be re-ordered; they are executed from top to bottom.
#
# If you remove a line here THAT COMMIT WILL BE LOST.
#
# However, if you remove everything, the rebase will be aborted.
#
# Note that empty commits are commented out
Rebase script

- Commits in reverse order from git log
- Shows order in which they will be applied
Rebase workflow

1. Decide what to do with each commit
2. Edit commit lines with appropriate commands

p 2dab5cd Fixed typo in class name
e f754ce3 added the bottom info blocks
e 7a6524a Self-service mobile css

1. Save the file and exit
2. Git will automatically pick the needed commits and stop at the ones you want to reword or edit
Editing commits

- When “edit” is chosen, Git will stop at the needed commits and let you choose what to do
Editing commits: choosing what to do

Stopped at f754ce3d9abbea2f1bd187d7d019aed842a80d5b... added the bottom info blocks
You can amend the commit now, with

    git commit --amend

Once you are satisfied with your changes, run

    git rebase --continue
Editing commits: choosing what to do

Stopped at f754ce3d9abbea2f1bd187d7d019aed842a80d5b... added
the bottom info blocks

You can amend the commit now, with

    git commit --amend

Once you are satisfied with your changes, run

    git rebase --continue
● Rebasing will change the SHA-1 hash of all the affected commits and everything that comes after them
● Generally you should only do it on local changes
● Local changes can be forced onto a remote:

$ git push -f <remote> <branch>

● OVERWRITES remote with local changes
● Potentially final and dangerous
● Only do if you are 100% sure what you are doing
Reordering commits

- Change the order of the lines in the rebase script to what you want
- Git will apply the commits in the provided order
Cancelling a rebase

- At any point, a rebase action can be completely cancelled

$ git rebase --abort
Practical session

- Notes on last practical session
- Browsing and searching history
- Restoring files
- Changing commit history
References


https://git-scm.com/docs/

https://en.wikipedia.org/wiki/SHA-1
