

Sharpening Your Git Skills



Matt Dickenson
September 15, 2016

 Basic Git workflow

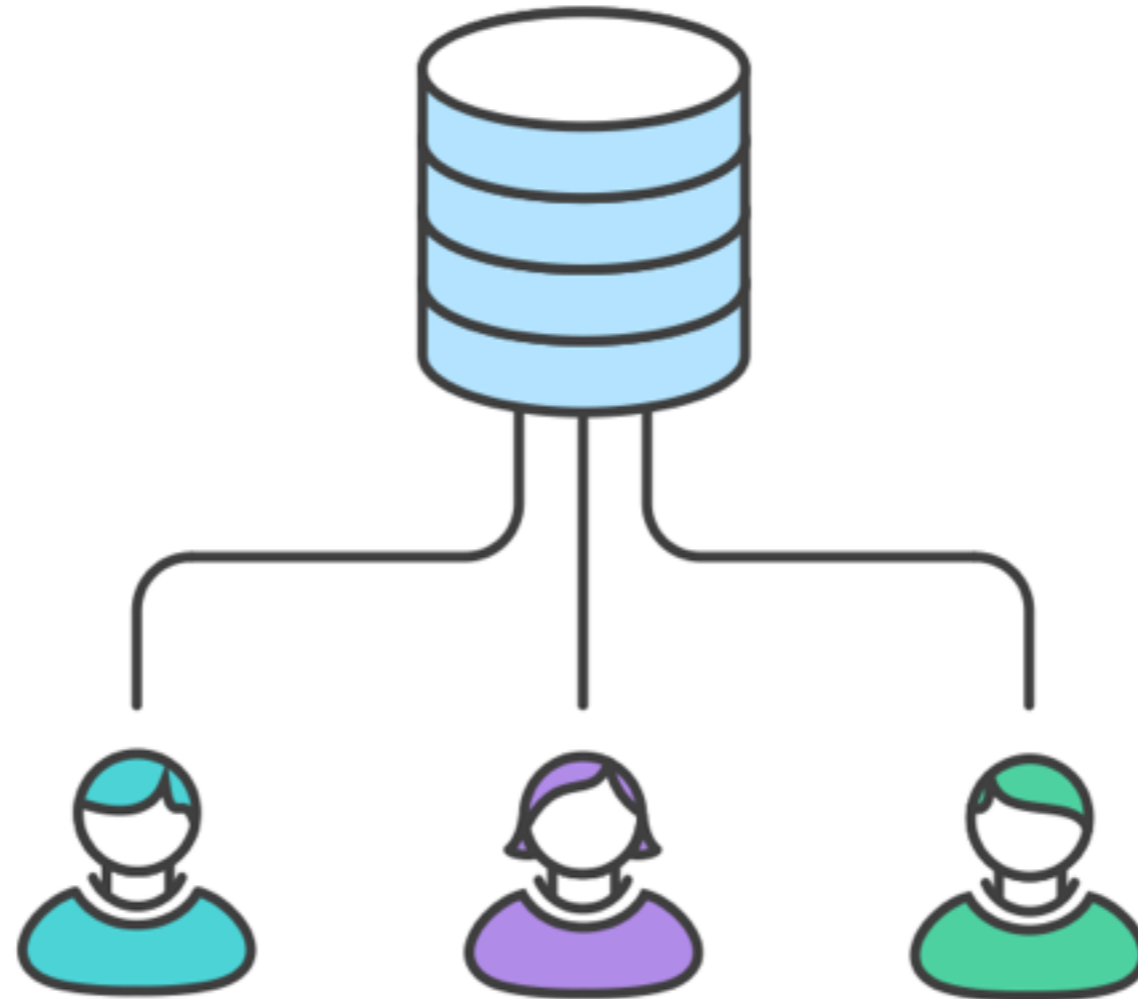
 Additional tools

 Working with Git history

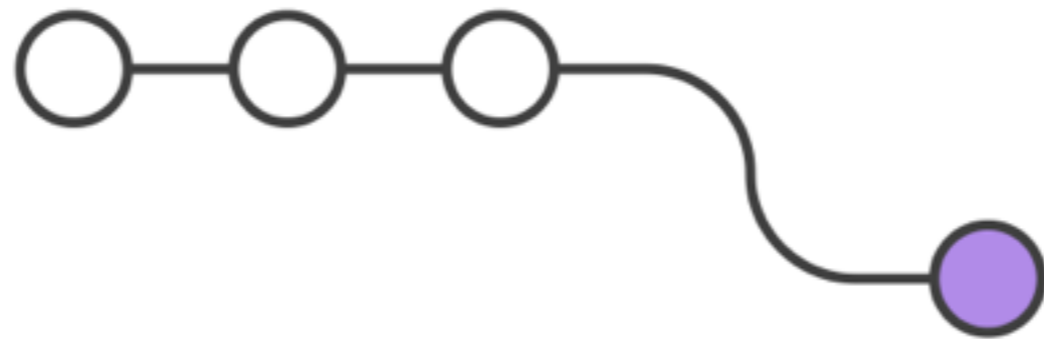
```
git init
```



git clone



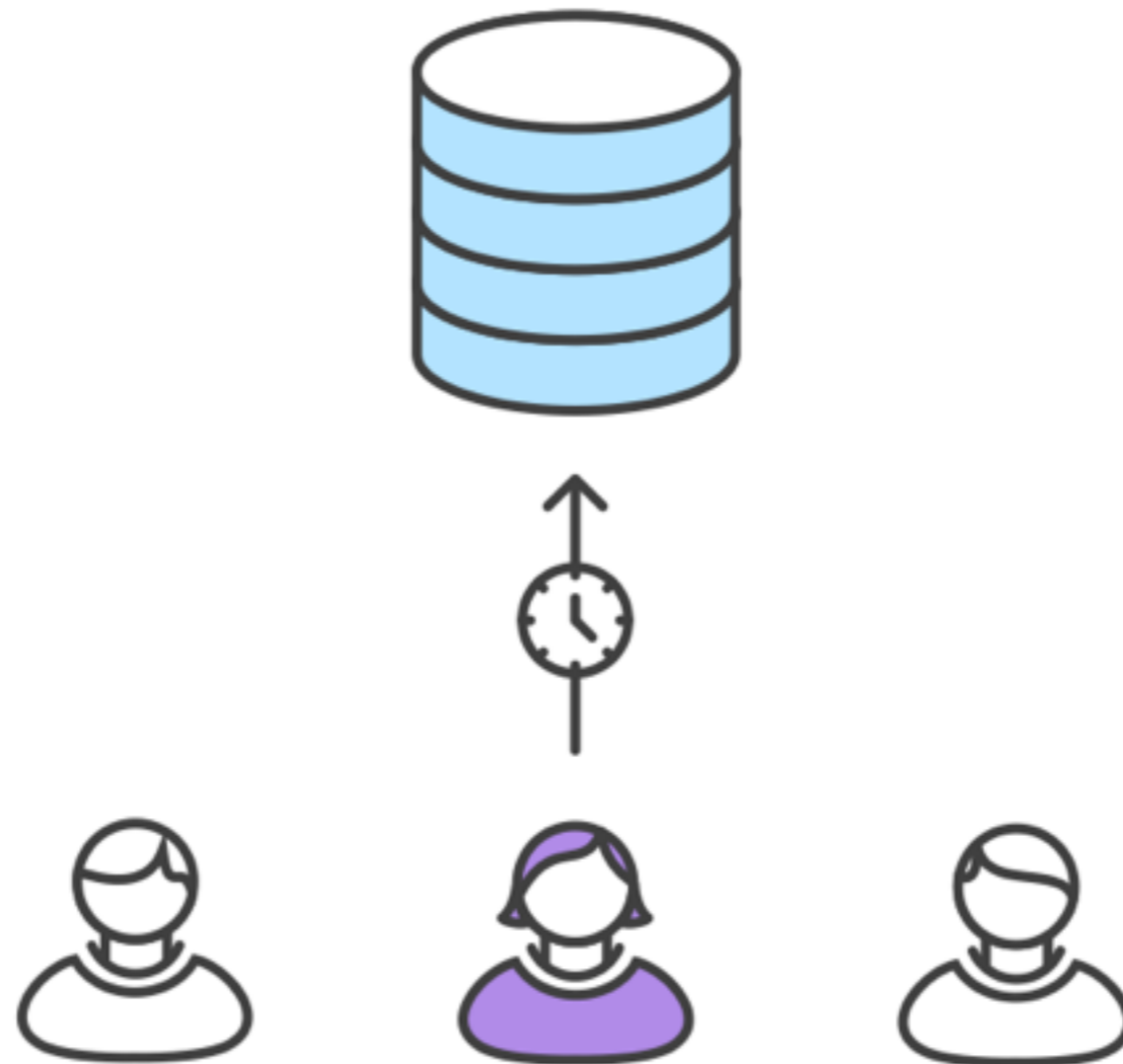
```
git checkout -b my-feature
```



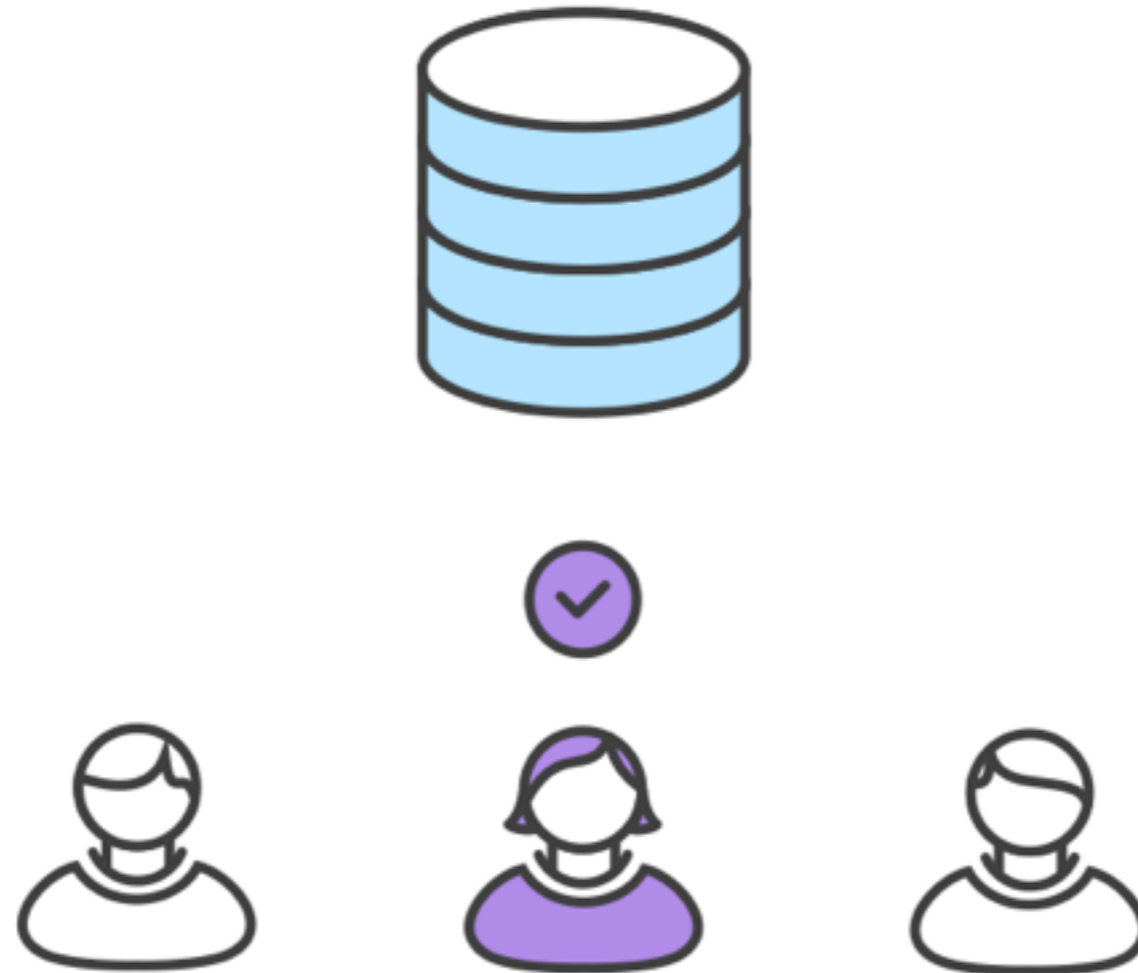
git status

git add

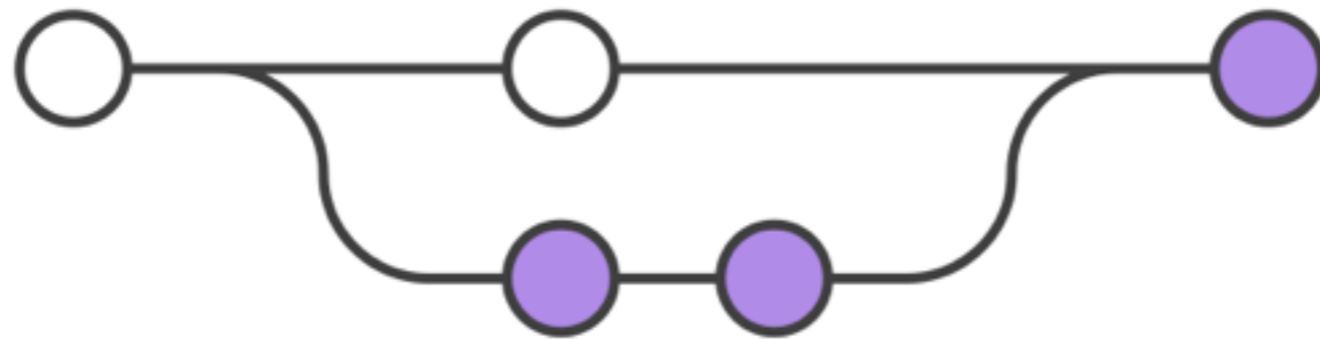
git commit

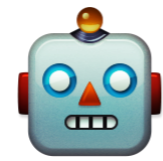


git push

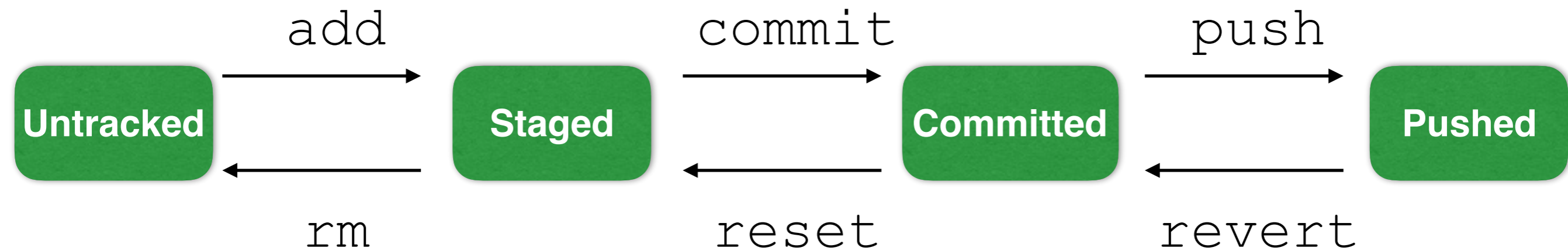


git merge





State Machine





Snapshotting

`clone`

create a local copy of a repository

`add`

update the index with current changes

`status`

show difference from latest commit

`diff`

compare two commits (or commit & working dir)



Snapshotting

`commit` save the current change set in a commit

`reset` go back to a specified state

`rm` remove a file from the index (stop tracking)

`mv` move/rename a file/directory/symlink



Branching and Merging

`branch`

list/create/delete branches

`checkout`

change the current branch (or restore a file)

`merge`

combine 2+ histories together

`log`

show history of a repo (or file)

`stash`

record current state, go back to clean working dir

`tag`

create/list/delete a GPG-signed object



Sharing and Updating

`fetch`

download from another (remote) repository

`pull`

fetch and merge (or rebase)

`push`

update remote repository

`remote`

manage which repositories you're tracking



Debugging

`bisect`

binary search to find which commit introduced a bug

`blame`

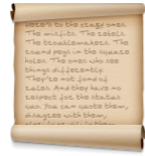
which author & commit changed each line of file

`grep`

show lines matching a pattern

`cherry-pick`

apply changes from existing commits



Working with History

`apply`

read a patch (diff output) and update files

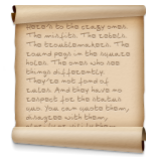
`rebase`

reapply commits on top of another base

`revert`

undo the changes in a commit

`filter-branch` rewrite a branch



Schools of Thought on History

School 1

“It’s a record of everything that happened”

It’s a historical document, valuable in its own right, and shouldn’t be tampered with. From this angle, changing the commit history is almost blasphemous; you’re lying about what actually transpired. So what if there was a messy series of merge commits? That’s how it happened, and the repository should preserve that for posterity.

Commits from School 1

```
fix some tests
```

```
Just a test.
```

School 2

“It’s the story of how your project was made”

You wouldn’t publish the first draft of a book, and the manual for how to maintain your software deserves careful editing. This is the camp that uses tools like rebase and filter-branch to tell the story in the way that’s best for future readers.

Commits from School 2

Convert module for
Spark pipeline.

Added utility to store
metrics in Cassandra.

School 1

- Pros
 - Easy, low barrier to entry
 - Records “what really happened”
- Cons
 - Different levels of granularity
 - Lots of commits
 - More to wade through when reading history

School 2

- Pros
 - Clean, meaningful history
 - Communicates purpose and intent
- Cons
 - Takes more time to write
 - Rewriting history has risks
 - Commits may have lots of code

	COMMENT	DATE
○	CREATED MAIN LOOP & TIMING CONTROL	14 HOURS AGO
○	ENABLED CONFIG FILE PARSING	9 HOURS AGO
○	MISC BUGFIXES	5 HOURS AGO
○	CODE ADDITIONS/EDITS	4 HOURS AGO
○	MORE CODE	4 HOURS AGO
○	HERE HAVE CODE	4 HOURS AGO
○	AAAAAAAAA	3 HOURS AGO
○	ADKFJSLKDFJSDKLFJ	3 HOURS AGO
○	MY HANDS ARE TYPING WORDS	2 HOURS AGO
○	HAAAAAAAAAANDS	2 HOURS AGO

AS A PROJECT DRAGS ON, MY GIT COMMIT MESSAGES GET LESS AND LESS INFORMATIVE.

Rewriting History

```
aa12ed5 Add c.txt  
d478c50 Add b.txt  
337d339 Add a.txt  
60d706a Initial commit
```


Rewriting History

```
pick 337d339 Add a.txt  
pick d478c50 Add b.txt  
pick aa12ed5 Add c.txt
```



```
pick 337d339 Add a.txt  
squash d478c50 Add b.txt  
squash aa12ed5 Add c.txt
```

```
git rebase -i HEAD~3
```

Rewriting History

```
# This is a combination of 3 commits.  
# The first commit's message is:  
Add a.txt  
  
# This is the 2nd commit message:  
  
Add b.txt  
  
# This is the 3rd commit message:  
  
Add c.txt
```

Rewriting History

```
af61b75 Add a.txt, b.txt, and c.txt.  
60d706a Initial commit
```

Rewriting History

```
af61b75 Add a.txt, b.txt, and c.txt.  
60d706a Initial commit
```

```
git commit --amend
```

Rewriting History

```
aa12ed5 Add c.txt  
d478c50 Add b.txt  
337d339 Add a.txt  
60d706a Initial commit
```



```
f210d94 Add text files for first three  
letters of alphabet.  
60d706a Initial commit
```

Questions for Commit Messages

1. Why is this change necessary?
2. How does it address the issue?
3. What side effects does this change have?



References

- Git book: git-scm.com/book
- Git workflows: atlassian.com/git/tutorials/comparing-workflows
- More on history: blimmer.github.io/1up-git-skills-talk

If all else fails...

