Intermediate Git

Day 2: Ignoring Files and Fixing Mistakes

Raman A. Shah

Use .gitignore for . . .

. . . secrets.

Use .gitignore for . . .

. . . local configuration.

Use .gitignore for . . .

. . . large unnecessary files.

Use `.gitignore` for . . .

. . . junk from your operating system and development environment.

“Glob patterns”

Ignore any file ending with `.txt`:

```
*.txt
```

But don’t ignore `IMPORTANT.txt`:

```
!IMPORTANT.txt
```

Ignore `temp0.out`, . . . , `temp9.out`, but not `tempa.out`:

```
temp[0-9].out
```

Ignore `temp0.out`, . . . , `temp9.out`, and also `tempa.out` (but not `temp_a.out`):

```
temp?.out
```

"Glob patterns," continued

Ignore file JUNK.tmp anywhere in the repo:

```
JUNK.tmp
```

Ignore file JUNK.tmp in the root directory, but don't ignore subdir/JUNK.tmp:

```
/JUNK.tmp
```

Ignore all files in the subdirectory subdir:

```
subdir/
```

Ignore log/foo.log but not foo.log in the root directory:

```
log/*.log
```

Initializing a Git repository

Download a compressed example of a messy project not under version control:

```
git clone https://github.com/\
ramanshah/intermediate_git.git
```

Pull it out of the course repo and decompress it into a directory:

```
cp ./intermediate_git/day2/sample.tgz .
tar xzvf sample.tgz
cd sample
```

Initializing a Git repository

From the directory that you're hoping to turn into a Git repository:

```
git init
```

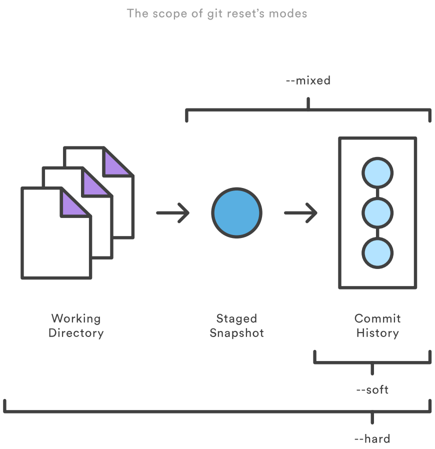
Survey the candidates to put under version control:

```
git status
```

Survey the files within a subdirectory:

```
git ls-files --others \
  --exclude-standard [subdir]
```

Scopes of `git reset`



`--mixed` is the default.

Atlassian, *Reset, Checkout, and Revert*.
CC-BY Australia.
https://www.atlassian.com/git/tutorials/

Fixing common mistakes

Git gives you anchors!

Uncommitted mistake #1

I screwed everything up but didn't commit. How do I return to my most recent commit?

```
git reset --hard HEAD
```

Uncommitted mistake #2

I modified one file incorrectly but didn't commit. Can I restore just that one file?

```
git checkout HEAD [path]
```

Committed mistake #1

I forgot to ignore something, and now it has changed and is chasing me around.

First, mark it for deletion in the staging area only:

```
git rm --cached [path]
```

Then commit the deletion:

```
git commit
```

Then fix your `.gitignore` and commit the changes to `.gitignore`.

Committed mistake #2

My most recent commit is half-baked. Can I bake it better?

Stage your corrections:

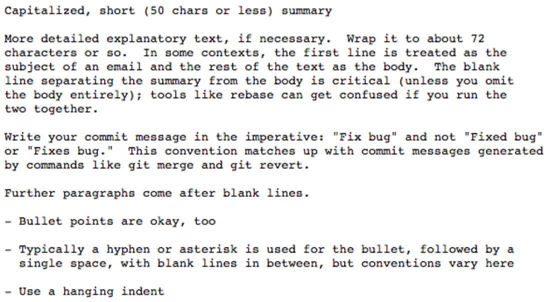
```
git add [path]
```

Then commit with a fresh message:

```
git commit --amend
```

Commit message style guide

- First line: $\leq$ 50 columns, imperative mood.
- Second line: blank.
- Subsequently: paragraph form, 72 columns, blank line between paragraphs.



```
Capitalized, short (50 chars or less) summary

More detailed explanatory text, if necessary.  Wrap it to about 72
characters or so.  In some contexts, the first line is treated as the
subject of an email and the rest of the text as the body.  The blank
line separating the summary from the body is critical (unless you omit
the body entirely); tools like rebase can get confused if you run the
two together.

Write your commit message in the imperative: "Fix bug" and not "Fixed bug"
or "Fixes bug."  This convention matches up with commit messages generated
by commands like git merge and git revert.

Further paragraphs come after blank lines.

- Bullet points are okay, too

- Typically a hyphen or asterisk is used for the bullet, followed by a
  single space, with blank lines in between, but conventions vary here

- Use a hanging indent
```

Tim Pope, *A Note About Git Commit Messages*

Committed mistake #3

I wish the last commit never happened.
Can I throw it away?

```
git reset --hard HEAD~1
```

Committed mistake #4

I wish the last commit never happened, but I want to keep the current stuff in the working tree.

```
git reset HEAD~1
```

The Golden Rule of Git

Never modify history that someone else has seen.

In public: git revert

Instead of deleting history with `git reset`, make *new* commits that reverse the damage done by a specified commit:

```
git revert HEAD~1
```