

Git and GitHub

Versioning

"FINAL".doc



↖ FINAL.doc!



↖ FINAL_rev.2.doc

Versioning



FINAL_rev.6.COMMENTS.doc



FINAL_rev.8.comments5.
CORRECTIONS.doc

Versioning

JORGE CHAM © 2012



FINAL_rev.18.comments7.
corrections9.MORE.30.doc

FINAL_rev.22.comments49.
corrections.10.#@\$%WHYDID
ICOMETOGRADSCHOOL?????.doc

Git vs Google Docs

	<i>saves current state ...</i>	<i>keeps track of ...</i>
Git	on request	directory tree
Google Docs	automatically	single file

The screenshot displays the Google Docs interface. On the left, a document titled 'Today, 1:45 PM' is open, showing a text editor with a highlighted sentence: 'Here I have added some text.' Below this, a section titled 'What Is Google Docs' is visible, containing text about Google Docs being an online word processor. On the right, the 'Version history' sidebar is open, showing a list of document versions. The current version is 'March 9, 1:45 PM' by Tina Sieber. Below it, there is a 'First Draft' version from 'March 9, 1:43 PM' and another version from 'March 9, 12:57 PM', both by Tina Sieber. A toggle switch for 'Only show named versions' is visible at the top of the sidebar.

← Today, 1:45 PM

100%

Version history

Only show named versions

Today

- ▶ **March 9, 1:45 PM**
Current version
■ Tina Sieber
- First Draft
March 9, 1:43 PM
■ Tina Sieber
- March 9, 12:57 PM
■ Tina Sieber

-Here I have added some text.

What Is Google Docs

Google Docs is Google's online word processor. What's its competitor, Microsoft Word, are its collaborative features. documents across platforms and work on them together in. Your collaborators don't even need a Google account to view or share with them.

Git: upsides

- collaboration
 - sharing
 - peer review
- backup
 - can go back to any version
 - simple and secure way to distribute to other machines

Git: downsides



Local workflow

- initialize a repository
 - `git init`

Working copy	Staging area	Commits
my-file.txt data .gitignore		


Local workflow

- initialize a repository
 - `git init`
- add files to staging area (where you prepare the next commit)
 - `git add my-file.txt`

Working copy	Staging area	Commits
my-file.txt data .gitignore	my-file.txt	

Local workflow

- initialize a repository
 - `git init`
- add files to staging area (where you prepare the next commit)
 - `git add my-file.txt`
- create a commit
 - `git commit -m 'change this and that'`

Working copy	Staging area	Commits
my-file.txt data .gitignore		<div>my-file.txt</div> 3c4a.. change this...

Diff

A shorthand for “differences”. Usually between working copy and the last commit.

‘-’ means line removed

‘+’ means line added

A sample from course materials repo:

```
``pull`` will do a merge for you
```

```
- ``git checkout --theirs``
```

```
+ ``git checkout --theirs {file path}``
```

```
in conflict, choose the version from the remote branch
```

Branches

- branches help you manage multiple versions of your code
- important for programmers, but we can't ignore it completely
- we'll be using one local branch called **master**
- on a remote, there is a twin branch, **origin/master**

GitHub

“Social part of programming”



Pushing to a remote [server]

- add a remote
 - `git remote add origin {where}`
- first push - also create a remote branch from the local one
 - `git push -u origin local-branch`
- push - upload commits
 - `git push`



Pulling from a remote

- download commits and merge into working copy
 - `git pull`

Pulling from a remote

- download commits and merge into working copy
 - `git pull`
- simple as that, unless there is a **merge conflict**



Resolving conflicts - binary files

- check what's conflicted
 - `git status`
- binary files are simple, choose either the incoming (theirs) or ours
 - `git checkout --ours binary-file.jpg`
 - `git add binary-file.jpg`
- commit the resolution, without any message
 - `git commit`
 - recently available: `git merge --continue`

Resolving conflicts - text files

- check what's conflicted
 - `git status`
- text files need to be edited
 - `nano my-file.txt`
 - `git add my-file.txt`
- commit the resolution, without any message
 - `git commit`
 - recently available: `git merge --continue`

Resolving conflicts

In text files, if both branches modify the same line, a conflict has to be resolved by deleting all the unwanted text (also the marks).

Here are lines that are either unchanged from the common ancestor, or cleanly resolved because only one side changed, or cleanly resolved because both sides changed the same way.

```
<<<<<< yours:sample.txt
```

```
Conflict resolution is hard;  
let's go shopping.
```

```
=====
```

```
Git makes conflict resolution easy.
```

```
>>>>>> theirs:sample.txt
```

```
And here is another line that is cleanly resolved or unmodified.
```