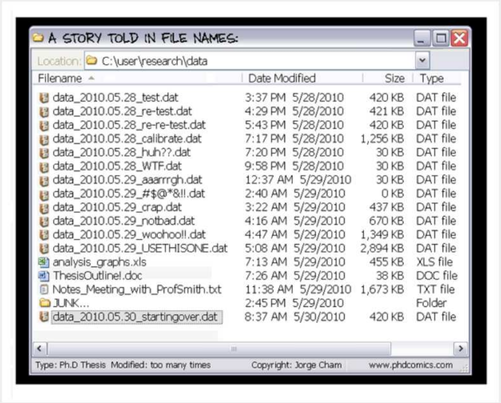
Jason Landini

University of Michigan
Ann Arbor, Michigan, United States

06 11 2025

Source Control - A Story Told in File Names

- Maintains history of files
- Tracks changes
- Facilitates collaboration
- Easy recovery to prior file states
- Safely experiment with backups



A STORY TOLD IN FILE NAMES:

Location: C:\user\research\data

Filename	Date Modified	Size	Type
data_2010.05.28_test.dat	3:37 PM 5/28/2010	420 KB	DAT file
data_2010.05.28_re-test.dat	4:29 PM 5/28/2010	421 KB	DAT file
data_2010.05.28_re-re-test.dat	5:43 PM 5/28/2010	420 KB	DAT file
data_2010.05.28_calibrate.dat	7:17 PM 5/28/2010	1,256 KB	DAT file
data_2010.05.28_huh??.dat	7:20 PM 5/28/2010	30 KB	DAT file
data_2010.05.28_WTF.dat	9:58 PM 5/28/2010	30 KB	DAT file
data_2010.05.29_aaarrrgh.dat	12:37 AM 5/29/2010	30 KB	DAT file
data_2010.05.29_#$@*&!!.dat	2:40 AM 5/29/2010	0 KB	DAT file
data_2010.05.29_crap.dat	3:22 AM 5/29/2010	437 KB	DAT file
data_2010.05.29_notbad.dat	4:16 AM 5/29/2010	670 KB	DAT file
data_2010.05.29_woohoo!!.dat	4:47 AM 5/29/2010	1,349 KB	DAT file
data_2010.05.29_USETHISONE.dat	5:08 AM 5/29/2010	2,894 KB	DAT file
analysis_graphs.xls	7:13 AM 5/29/2010	455 KB	XLS file
ThesisOutline!.doc	7:26 AM 5/29/2010	38 KB	DOC file
Notes_Meeting_with_ProfSmith.txt	11:38 AM 5/29/2010	1,673 KB	TXT file
JUNK...	2:45 PM 5/29/2010		Folder
data_2010.05.30_startingover.dat	8:37 AM 5/30/2010	420 KB	DAT file

Type: Ph.D Thesis Modified: too many times Copyright: Jorge Cham www.phdcomics.com

Note on git tracking files

We can use git for things beyond code like media, documents, pdf storage, etc...

Source Control - Terminology

branch a divergence from the main trunk, allowing independent development without affecting the trunk

git commands

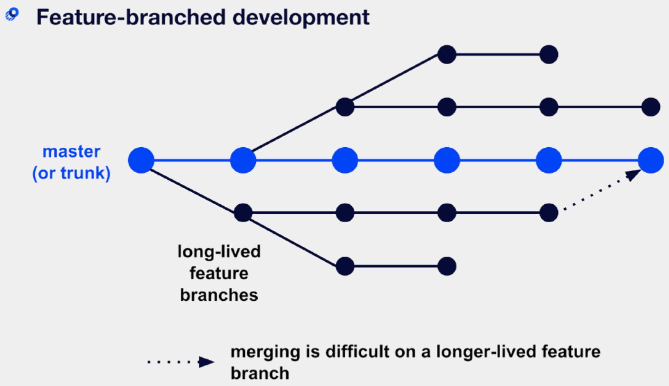
`git branch <branch>`	*Create a new branch*
`git checkout <branch>`	*Switch to the branch*
`git checkout -b <branch>`	*Create and switch to a branch*

integrating branches once a branch has matured, it should be merged back into the main trunk

git commands

`git merge`	*Combines history*
`git rebase`	*Rewrites history*
`git squash`	*Squashes new history to single point*

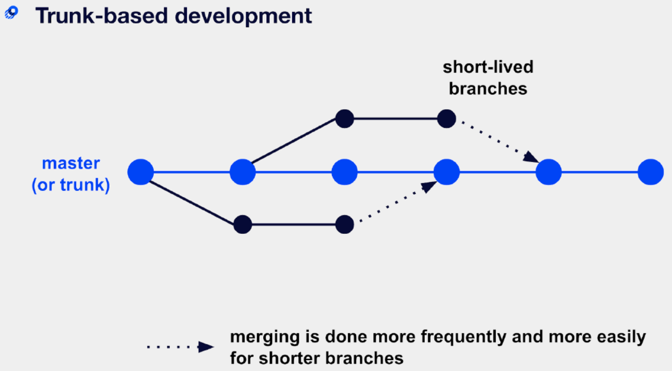
Source Control - Feature-based Development


Feature-branched development
master (or trunk)
long-lived feature branches
merging is difficult on a longer-lived feature branch

[1]

[1]https://www.optimizely.com

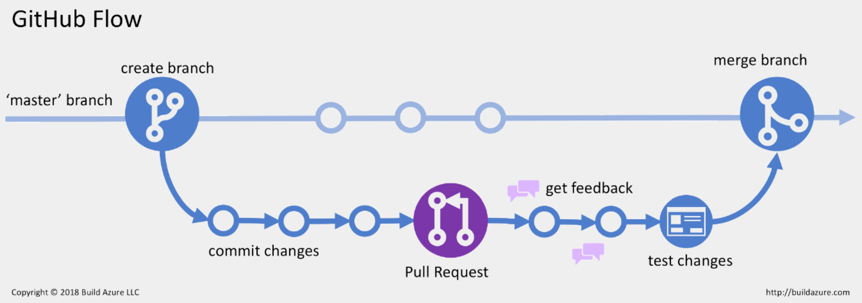
Source Control - Trunk-based Development


Trunk-based development
short-lived branches
master (or trunk)
merging is done more frequently and more easily for shorter branches

2

[2] https://www.optimizely.com

Source Control - Developer Workflow

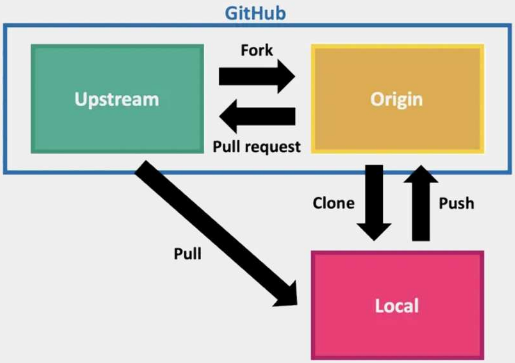


3

[3] https://buildazure.com

Source Control - Forks

- Forks copy a repository
- Avoids polluting a repository with feature branches
- Allows contribution to projects you don't own



[a]

[a]https://medium.com/@jacoblogan98/understanding-git-branching-5d01f3dda541

Source Control - Continuous Integration

Continuous Integration (CI) is a way to automatically test code during development.

Common workflows:

- Auto-formatting - ensures consistent style
- Tests - unit tests, regression tests, fuzz tests, etc..
- Static analysis - checks code quality against coding standards
- Build checks - verifies successful builds across systems
- Code coverage - reports how much code is covered by unit tests
- Documentation - ensures docs are up-to-date

CI on GitHub

GitHub Actions are **free** for public repositories!

Source Control - Etiquette

- Avoid large pull requests (>500 lines)
- Keep it simple. Avoid unrelated refactoring
- Follow the project's style guide
- Write tests when adding features
- Write meaningful pull request titles & descriptions
- Link related issues
- Create an issue if one doesn't exist
- Large changes are better discussed in an issue or discussion rather than a pull request
- Ask for review only when CI is passing
- Outside conversations should be documented in GitHub

Demo - Setting up git & GitHub

Install git (type `git` in a terminal to check)

Set your name and email address:

```
git config –global user.name ”<name>”
git config –global user.email ”<email>”
```

Create a GitHub account and a ssh-key (if you don’t have one) with `ssh-keygen`

Link your laptop’s ssh key to your GitHub account under `https://github.com/settings/keys`

Demo - Forking

Create a fork of

```
https://github.com/IPAM-ECH2025/PoissonBoltzmannIPAM2025
```

In your terminal clone your fork with

```
git clone
git@github.com:username/PoissonBoltzmannIPAM2025.git
```

Add an upstream with

```
git remote add upstream https://github.com/IPAM-
ECH2025/PoissonBoltzmannIPAM2025.git
```

You can verify your remotes with

```
git remote -v
```

Demo - Making a PR

On you local computer create a feature branch and make some changes with

```
git checkout -b fix_typos
```

make some changes to files

```
git add [files]
```

```
git commit -m ”<message>”
```

```
git push origin fix_typos
```

On GitHub go to your fork and create a pull request using the branch you just created. After review, it will get merged and you can delete your branch.