Intro to Git

Objectives

- Understand the necessity of version control
- Demonstrate the difference between Git and GitHub
- Explore the features of Git
- DEMO: construct a GitHub repository and address merge conflicts

Version Control

final.doc

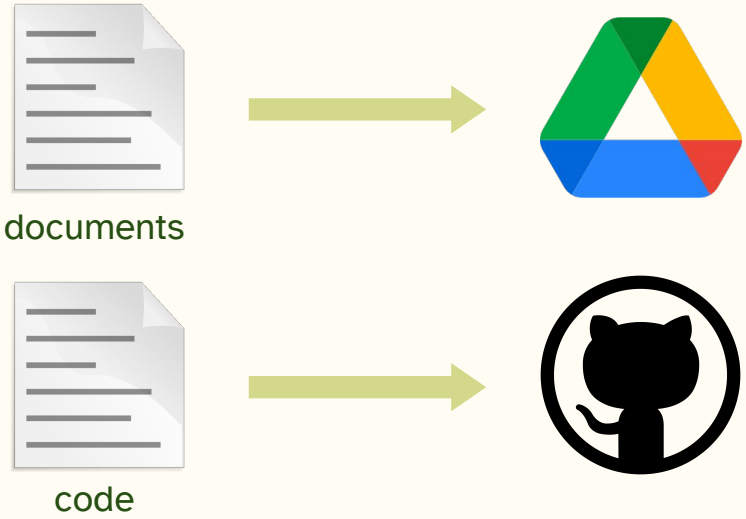
final_2.doc

actual_final.doc

FINAL.doc

LASTFINAL.doc

Version Control



Most software companies use some form of version control (not all use Git).

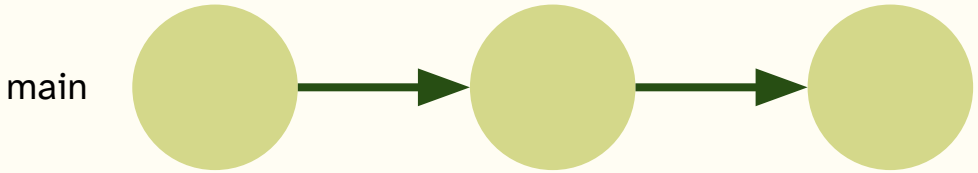
Benefits of Git

- Distributed
- Non-linear
- Graceful merges
- Github as a remote

Git Glossary

- **repository** (or **repo**): the storage of all your files
- **commit**: a record of your changes, along with the author and timestamp
- **pull**: retrieve updates from your repo
- **pull request**: a method of requesting that your changes be added to the repository
- **push:** add updates to your repo
- **remote**: a version of your project hosted elsewhere (usually the cloud)

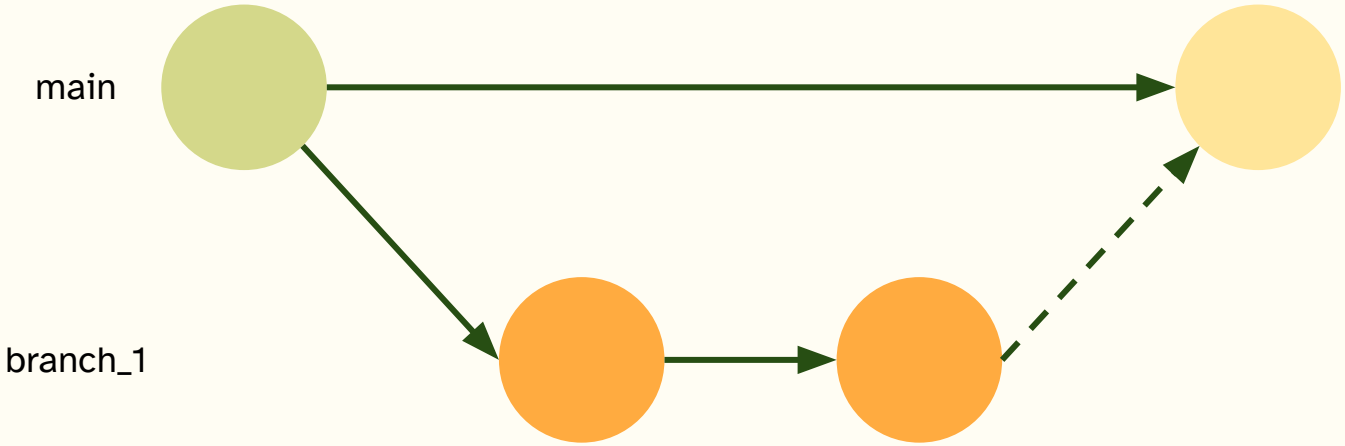
Tracking Changes



Using Github: Step 1

1. Create a remote repository on GitHub
2. Create or modify a file on your local machine
3. Add these updates to your commit
    a. Github creates a unique ID (hash) for each commit
4. Push your commit(s) up to your main branch

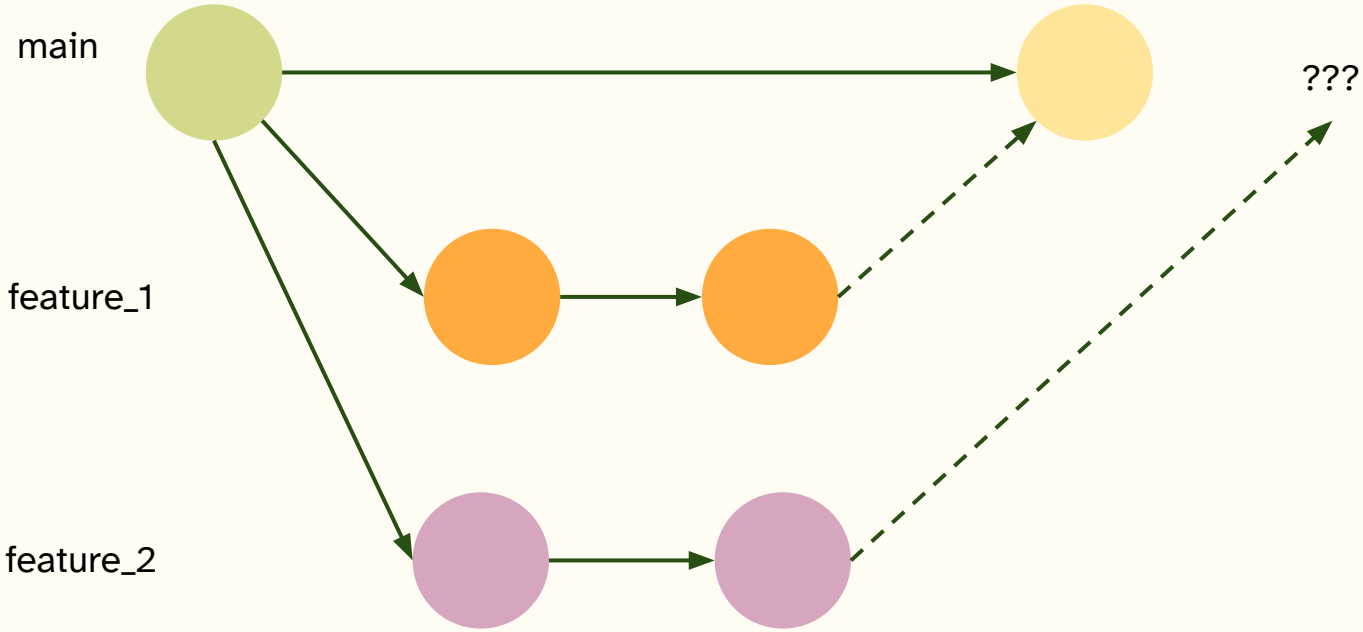
Tracking Changes



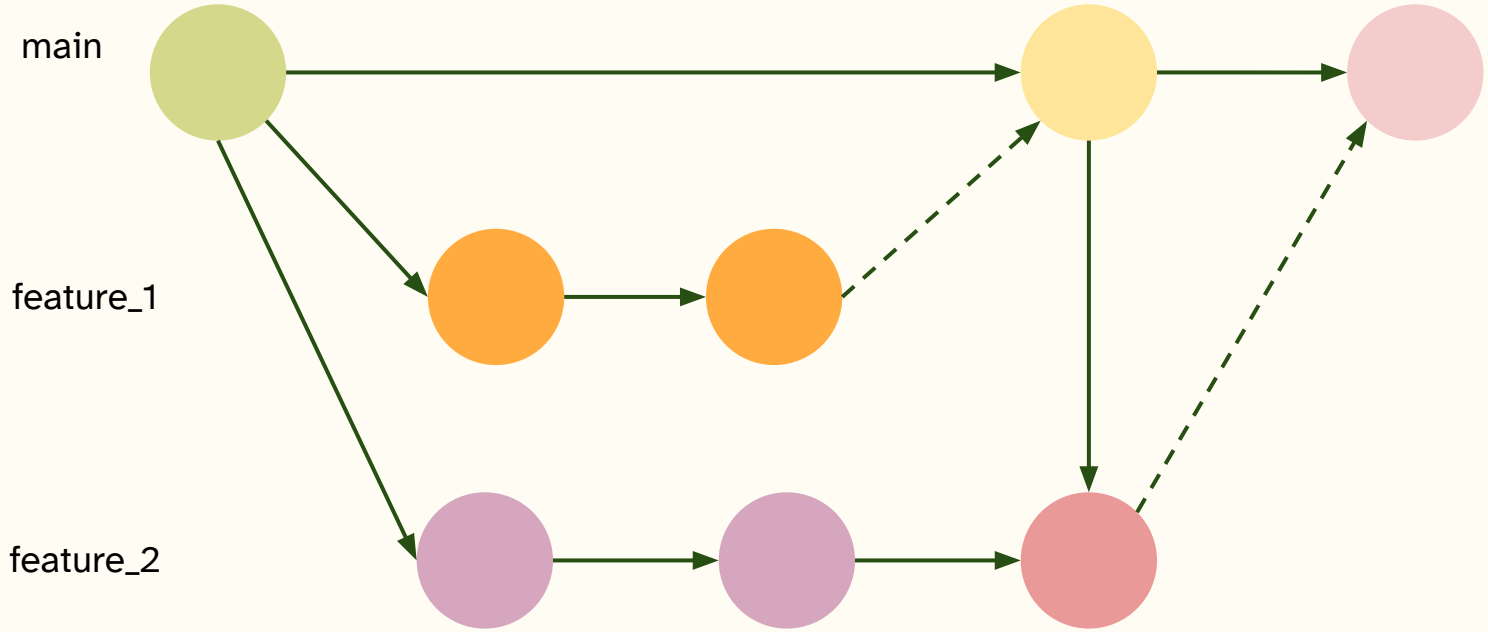
Using Github: Step 2

1. Create a remote repository on GitHub
2. Create or modify a file on your local machine on a **feature branch**
3. Add these updates to your commit
    a. Github creates a unique ID (hash) for each commit
4. Push your commit(s) up to the remote **feature branch**
5. Create a **pull request** from the GitHub site
6. If approved, you can **merge** your commits into the **main** branch

Tracking Changes



Tracking Changes



Using Github: Step 3

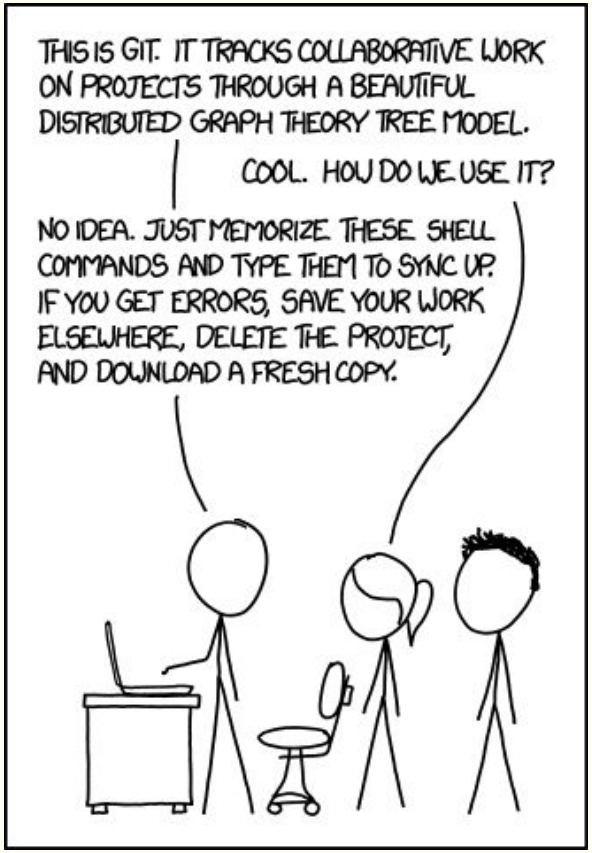
1. Create a remote repository on GitHub
2. Create or modify a file on your local machine
3. Add these updates to your commit
    a. Github creates a unique ID (hash) for each commit
4. Push your commit(s) up to the remote **feature branch**
5. If you cannot merge a **pull request** from the GitHub site automatically:
    a. Merge the **main** branch into your feature branch
    b. Resolve all conflicts
    c. Then, create a new commit and push it up to the remote feature branch
6. If approved, you can **merge** your commits into the **main** branch

Commands to Remember

- **git add**: adds your changes to the staging area
- **git commit**: creates a commit from your staging area
- **git status**: tells you what branch you're on and what files you've edited
- **git log**: shows the history of your repository
- **git push <remote> <local_branch>**: adds your changes to the remote
- **git pull <remote> <local_branch>**: receives your changes from the remote

Final Thoughts

1. Git docs are REALLY good
2. StackOverflow is your friend
3. Explore GitHub -- it has loads of insights and features



Please don't do this.

That's it!

Open floor for questions & curiosities.