Building software: Version control with Git

Data Sciences Institute

University of Toronto

Simeon Wong

Asking questions

- Zoom chat during class
  - Feel free to post and answer questions at any time
  - I will pause for questions occasionally, and review questions from the chat
- Pre- / Post-class office hours with Tong
- Email
  - simeonm.wong@mail.utoronto.ca
  - tong.su@mail.utoronto.ca

UNIVERSITY OF TORONTO

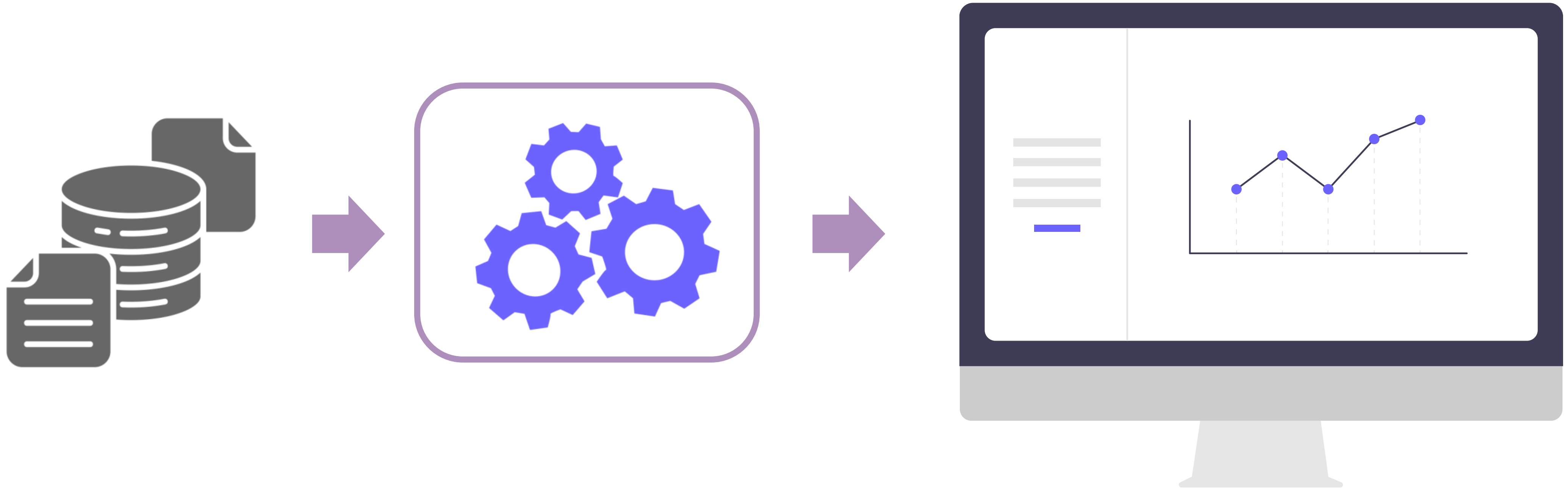
Course objective

How to write robust software in a team that we, our colleagues, and the public can trust and use with confidence.

Git: Branching

Alex's new data pipeline

- Alex is a data engineer at a mid-sized company working on a new data processing pipeline and BI dashboard module



Git: Branching

Alex’s new data pipeline

- Alex is a data engineer at a mid-sized company working on a new data processing pipeline and BI dashboard module

Git: Branching

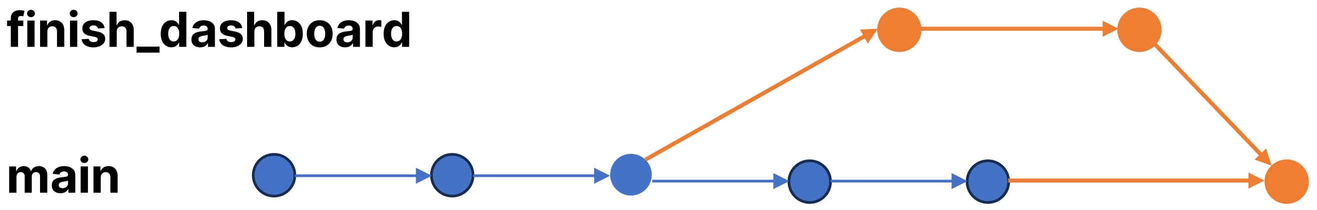
Alex’s new data pipeline

- Alex is a data engineer at a mid-sized company working on a new data processing pipeline and BI dashboard module
- Alex has a basic data pipeline and most of the BI module written
- Alex is currently working on expanding the data pipeline with more features. The expanded pipeline is not yet working, but……
- She has a big client meeting coming up and they want a demo!

Git: Branching

Alex’s new data pipeline

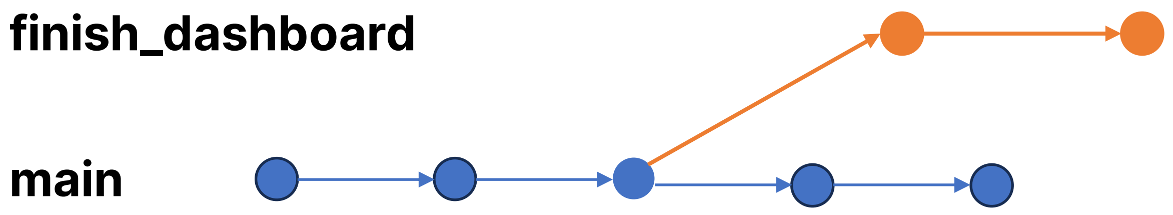
- Alex can use Git to go back to the last working state of her basic data pipeline
- Alex can finish up the BI module on another **branch**
- Present the amazing new BI module and wow then client
- Then **merge** her dashboard work back into the main branch incorporating both the in-progress pipeline and finished BI module



Git: Branching

$> Interactive live coding

Finish the dashboard on a separate branch.

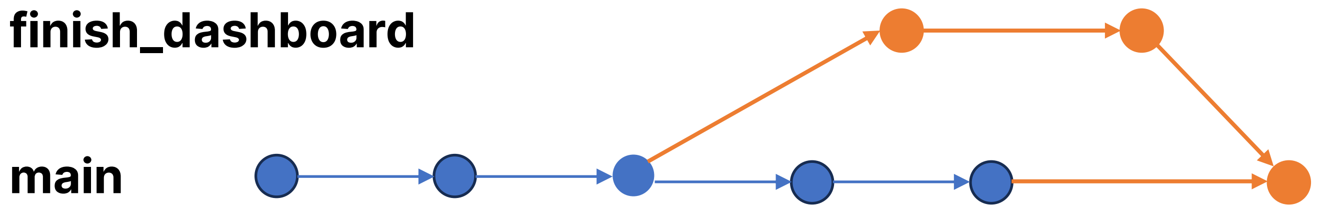


1. Clone https://github.com/dtxe/DSI_branch_demo
2. Switch to good commit
3. Finish the dashboard

Git: Branching

$> Interactive live coding

Merge the dashboard work into the main branch.



1. Switch to main
2. Merge finish_dashboard

Git: Branching

Tracking changes with Alex

- Follow along as Alex uses Git to simplify her work
  - Create a new branch from a commit `git switch`
  - Merge changes from another branch `git merge`

What questions do we have?

Git: Branching

Listing branches

- List branches in your repo with

```
git branch -v
```

$> Let's try it now!

Git: Branching

Deleting branches

- Delete branches in your repo with

```
git branch –d <branch name>
```

- Git will warn you if your branch contains work that hasn’t been incorporated into the main branch yet
    - But it is best practice to check before deleting anyways!

Pop-quiz: How do we check what commits are in a branch?

UNIVERSITY OF TORONTO

Git: Branching

Deleting branches

- Delete branches in your repo with

```
git branch -d <branch name>
```

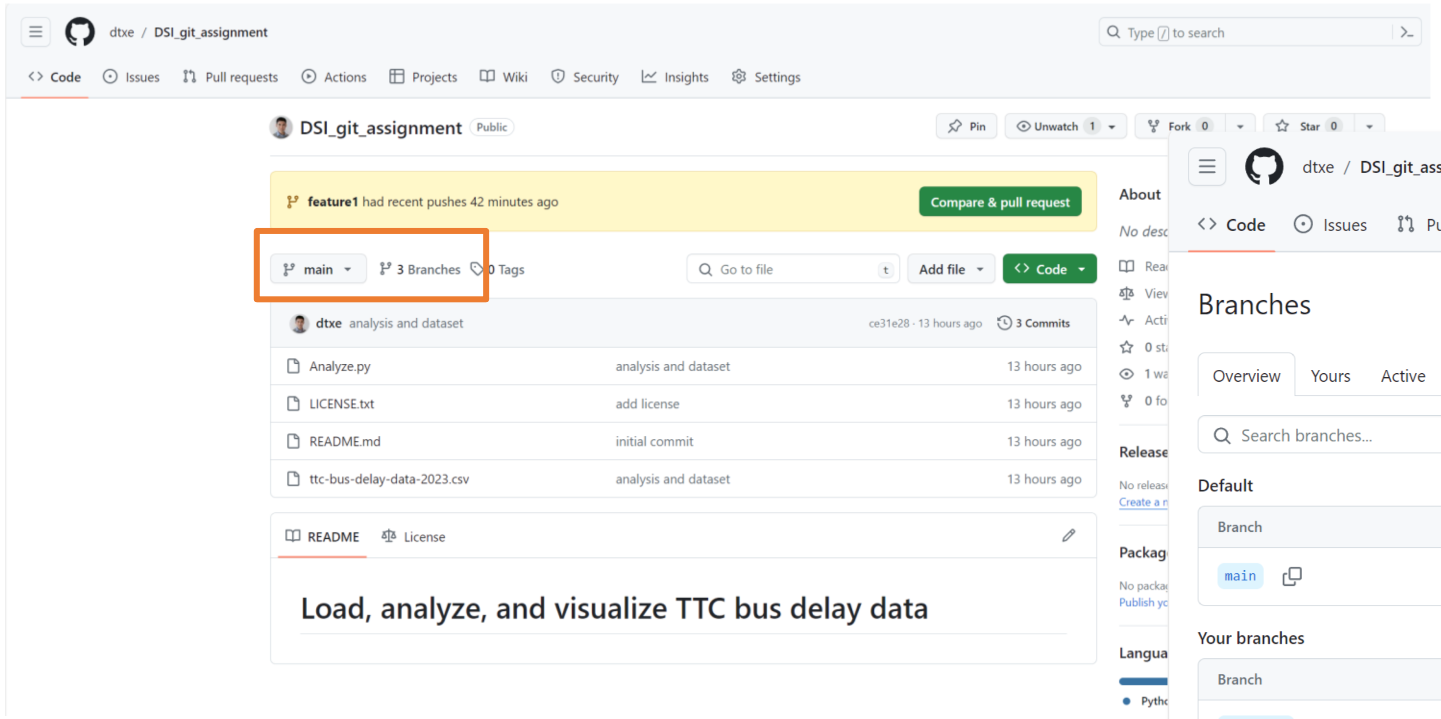
- Git will warn you if your branch contains work that hasn't been incorporated into the main branch yet
  - But it is best practice to check before deleting anyways!

$> **Let's try it now!**

- Create a branch, make a commit, try deleting, merge, try deleting again

Git: Branching

Branches on GitHub



dtxe / DSI_git_assignment

DSI_git_assignment Public

feature1 had recent pushes 42 minutes ago — Compare & pull request

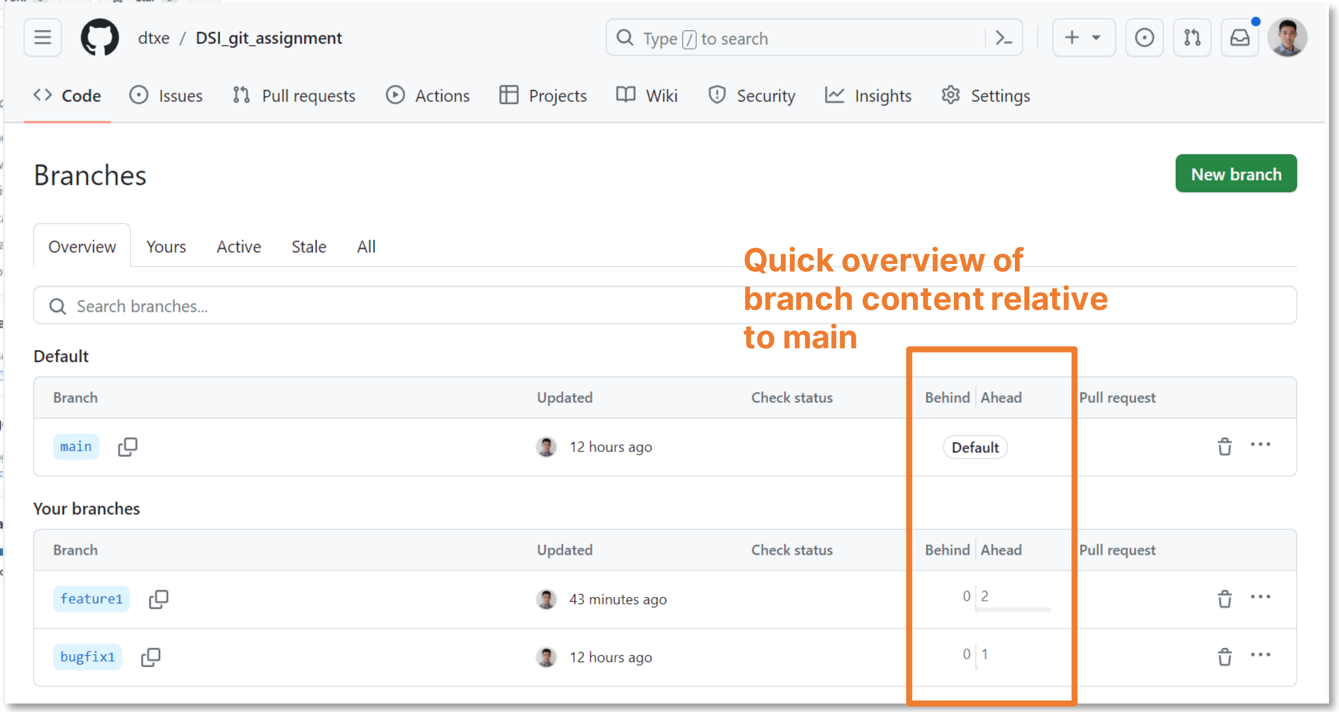
main | 3 Branches | 0 Tags

dtxe analysis and dataset — ce31e28 · 13 hours ago — 3 Commits

Analyze.py	analysis and dataset	13 hours ago
LICENSE.txt	add license	13 hours ago
README.md	initial commit	13 hours ago
ttc-bus-delay-data-2023.csv	analysis and dataset	13 hours ago

README | License

Load, analyze, and visualize TTC bus delay data



dtxe / DSI_git_assignment

Branches

Default

Branch	Updated	Check status	Behind | Ahead	Pull request
main	12 hours ago		Default	

Your branches

Branch	Updated	Check status	Behind | Ahead	Pull request
feature1	43 minutes ago		0 | 2	
bugfix1	12 hours ago		0 | 1	

Quick overview of branch content relative to main

$> Let’s try it now!

UNIVERSITY OF TORONTO

Git fetch

- Ask git to download from remote repositories
- Does not change your current working directory
- Enables subsequent merge from or switch to remote branches

```
            git fetch upstream newfeature
     THEN   git merge upstream/newfeature
     OR     git switch -c newfeature upstream/newfeature
```

Pull = Fetch + Merge

- The combined fetch and merge happens very often
- Combined into the verb pull

```
git pull upstream newfeature
```

essentially performs:

```
git fetch upstream newfeature

git merge upstream/newfeature
```

What questions do we have?

UNIVERSITY OF TORONTO

Git in VSCode

- Basic git commands are built-in with VSCode
  - Staging files, Commits, Branches

- View relationship between git commits intuitively with Git Graph

> $> **Let's try it now!**

What questions do we have?

Git: Branching

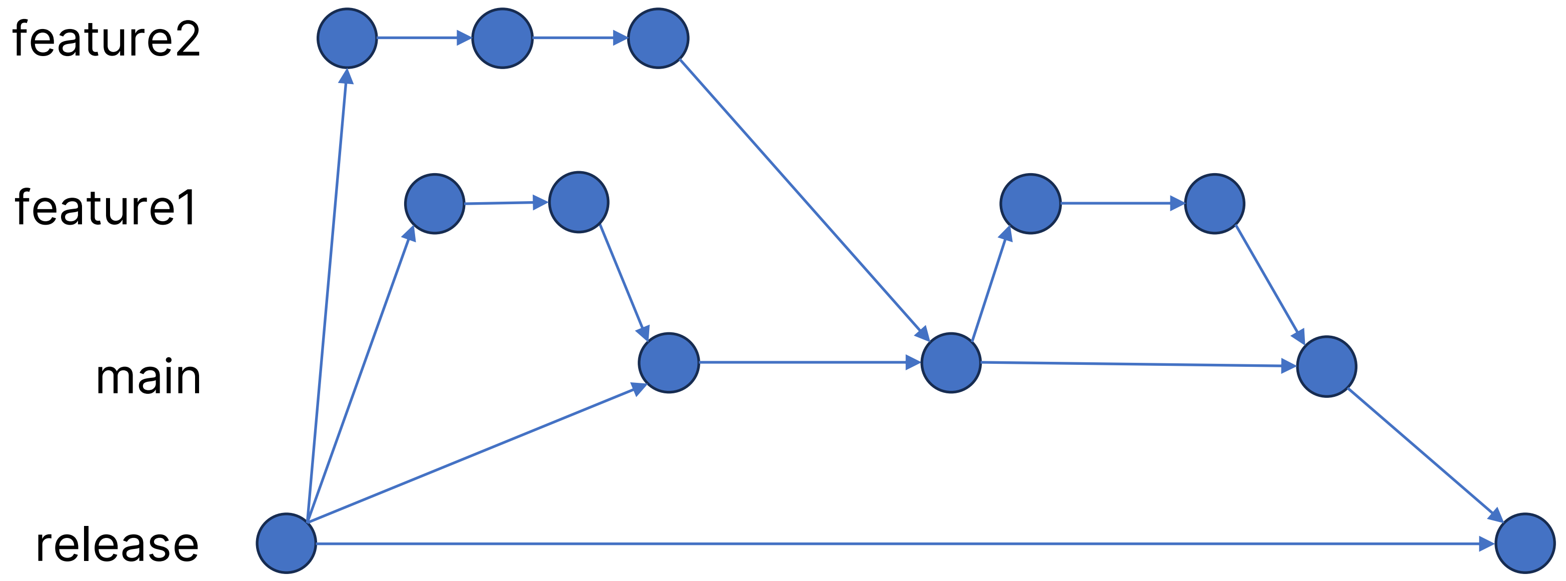
Everything is a branch

- Including forks!
- We can merge from local branches, forks, remote branches, etc...

Git: Branching

Branch workflows

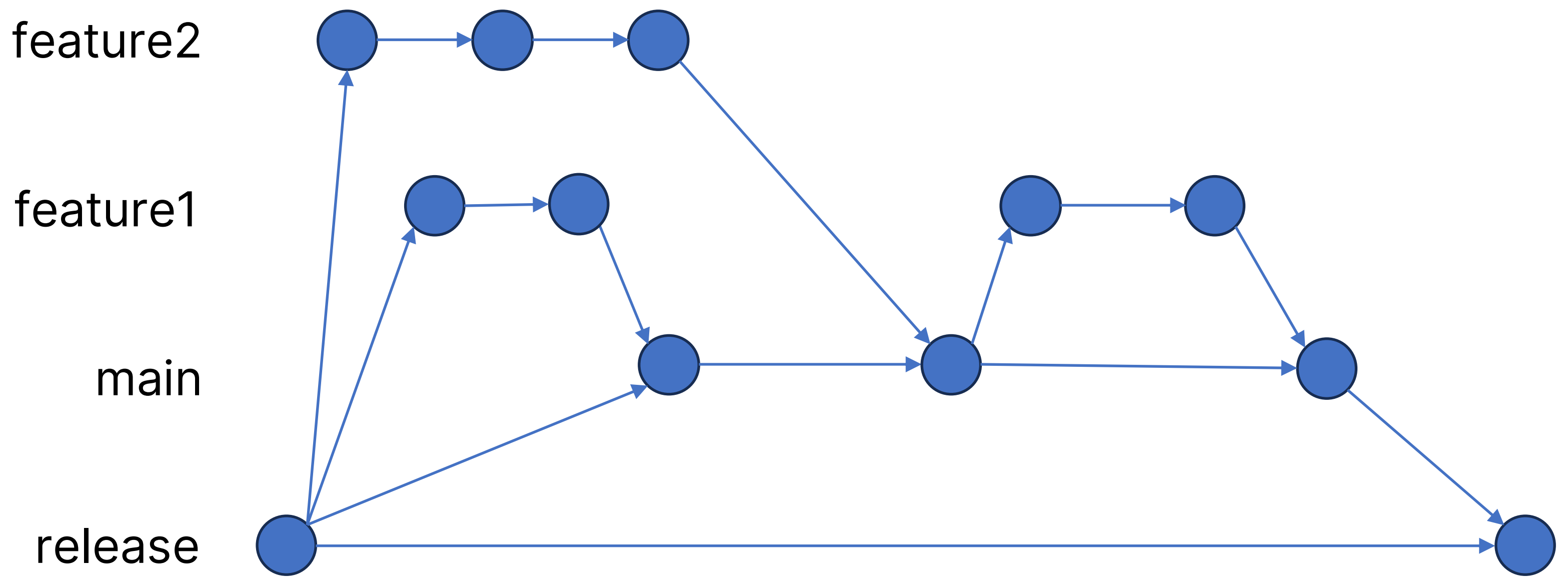
- Multiple long-running branches are helpful for large projects
- Features are developed in their own branches, based on release commits



Git: Branching

Branch workflows

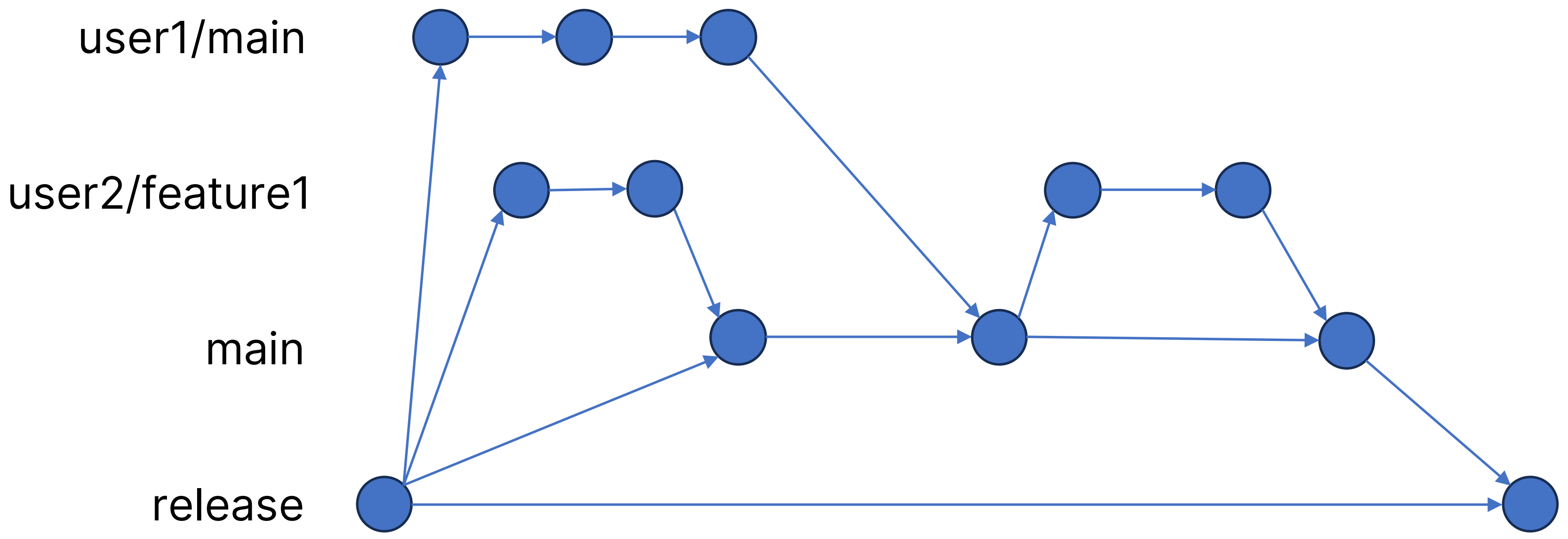
- Branches can have various levels of stability
    - Code can graduate/merge from **feature/topic** branches when stable
    - Features can be developed in parallel and merged into main



Git: Branching

Branch workflows

- Branches can be user forks too



What questions do we have?

UNIVERSITY OF TORONTO

Merge conflicts

- Code might be modified in both branches that are being merged
- Git does not know which change (or neither or both) should be kept
  - More recent is not always the right one!
- Git will alert you that you need to step in.

Git: Merge conflicts

$> Interactive live coding

1. `git switch -c feature1`
2. Edit `mycode`, commit
3. `git switch main; git switch -c feature2`
4. Edit `mycode`, commit
5. Merge `feature1`
6. Resolve conflict, complete merge

What questions do we have?

Git: Ignoring files

.gitignore

- Why?
  - Large data files, intermediate output, secret keys, etc...
- Defined in the `.gitignore` file
- Specify path with wildcards
- **Best practice:** use existing `.gitignore` templates

Course objective

How to write robust software in a team that we, our colleagues, and the public can trust and use with confidence.

Homework #2

- Due tomorrow before class
- Clone a repo, merge some branches, resolve a conflict
- This homework is also part of the Git Assignment

- Detailed instructions on the GitHub repo