Building software: Version control with Git

Data Sciences Institute
University of Toronto

Simeon Wong

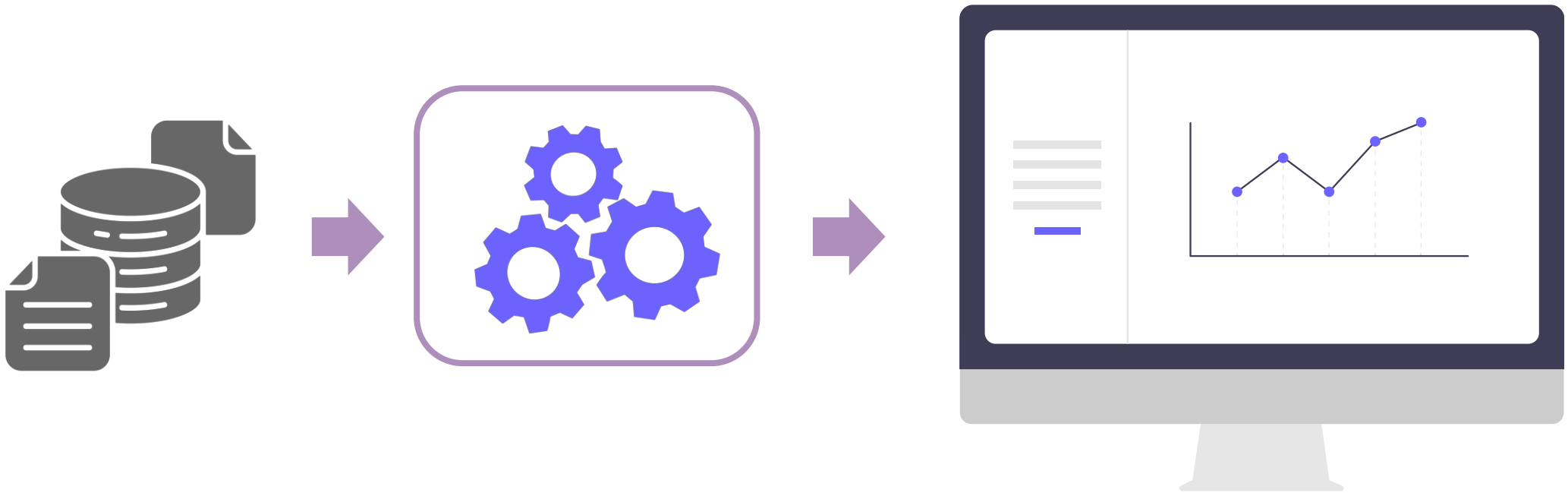
Course objective

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

Alex's new data pipeline

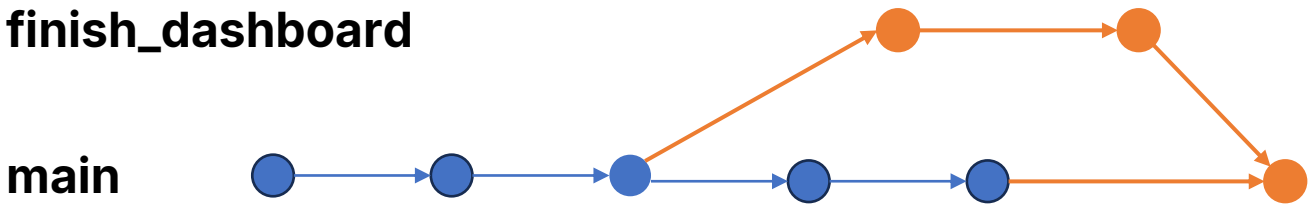
- Alex is a data engineer at a mid-sized company working on a new data processing pipeline and BI dashboard module

- 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

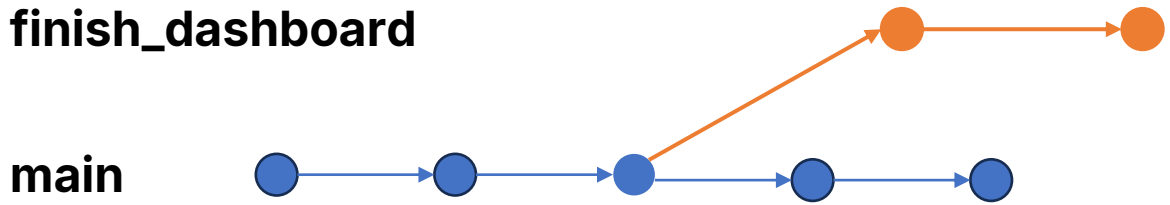
- Alex can use Git to go back to the last working state of her basic data pipeline
- Alex can finish up the BI module on another **branch**
- Present the amazing new BI module and wow then client
- Then **merge** her dashboard work back into the main branch incorporating both the in-progress pipeline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

```
git clone https://github.com/dtxe/DSI_branch_demo
```

2. Switch to good commit

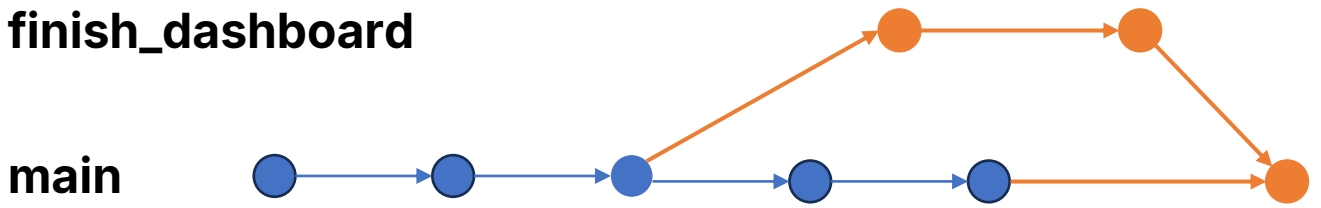
```
git switch -c finish_dashboard 6539845acab60c73ab50b65d58d9e39fd4a10119
```

3. Finish the dashboard

```
touch dashboard ;  git add -A ;  git commit -m “finish dashboard”
```

$> Interactive live coding

Merge the dashboard work into the main branch.



1. Switch to main

```
git switch main
```

2. Merge finish_dashboard

```
git merge finish_dashboard
```

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`

Questions?

Listing branches

- List branches in your repo with

```
git branch -v
```

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

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?

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

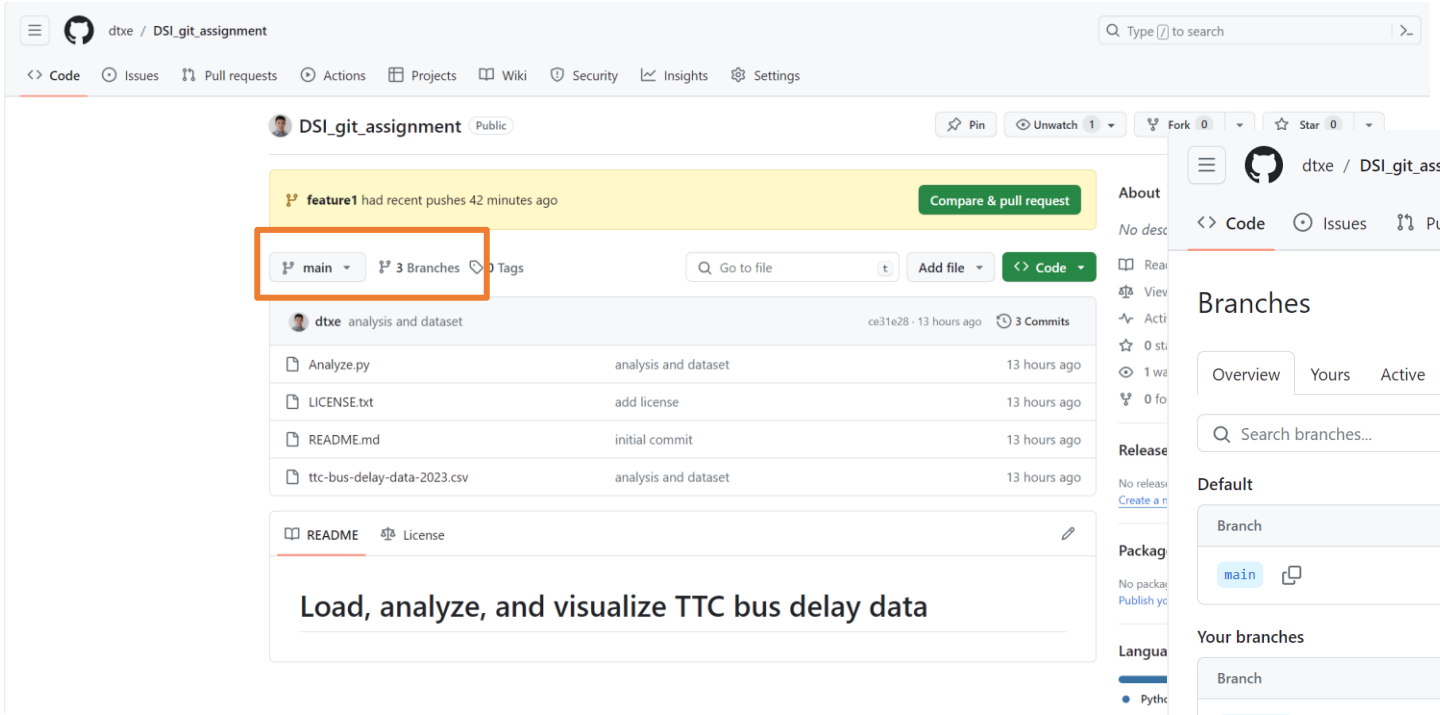
Git: Branching

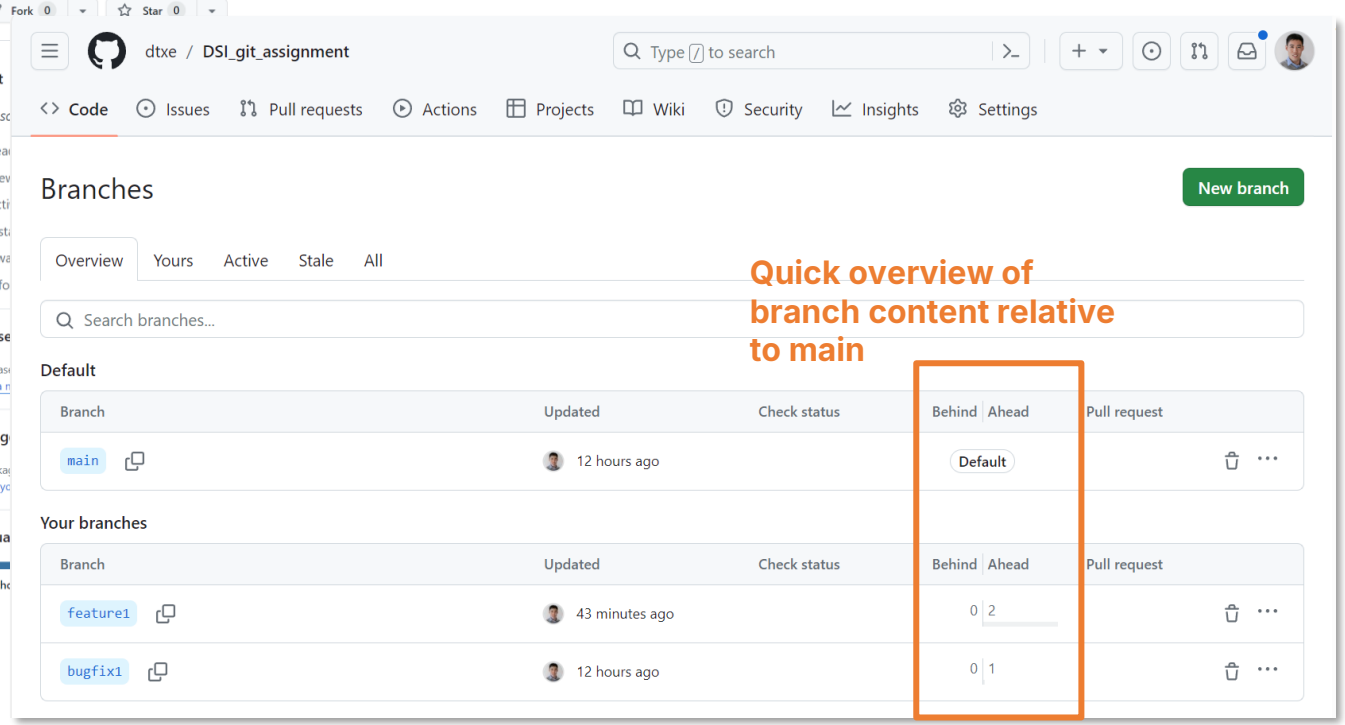
$> Deleting branches

```
git switch -c newbranch

touch newfile ; git add -A ; git commit -m “newfile”

git switch main

git branch -d newbranch

    >>> Error: branch is not fully merged

git merge newbranch

git branch -d newbranch

    >>> Success!
```

Git: Branching

Branches on GitHub





$> Let's try it now!

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
```

Questions?

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!**

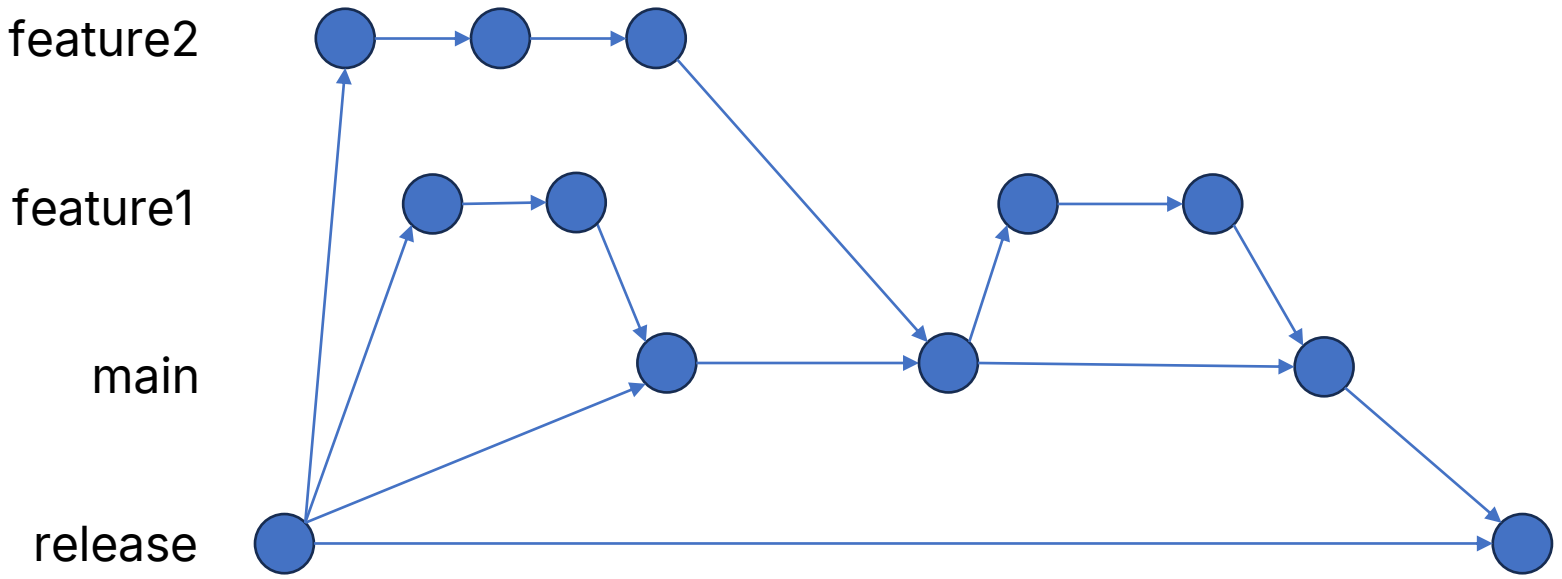
Questions?

Everything is a branch

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

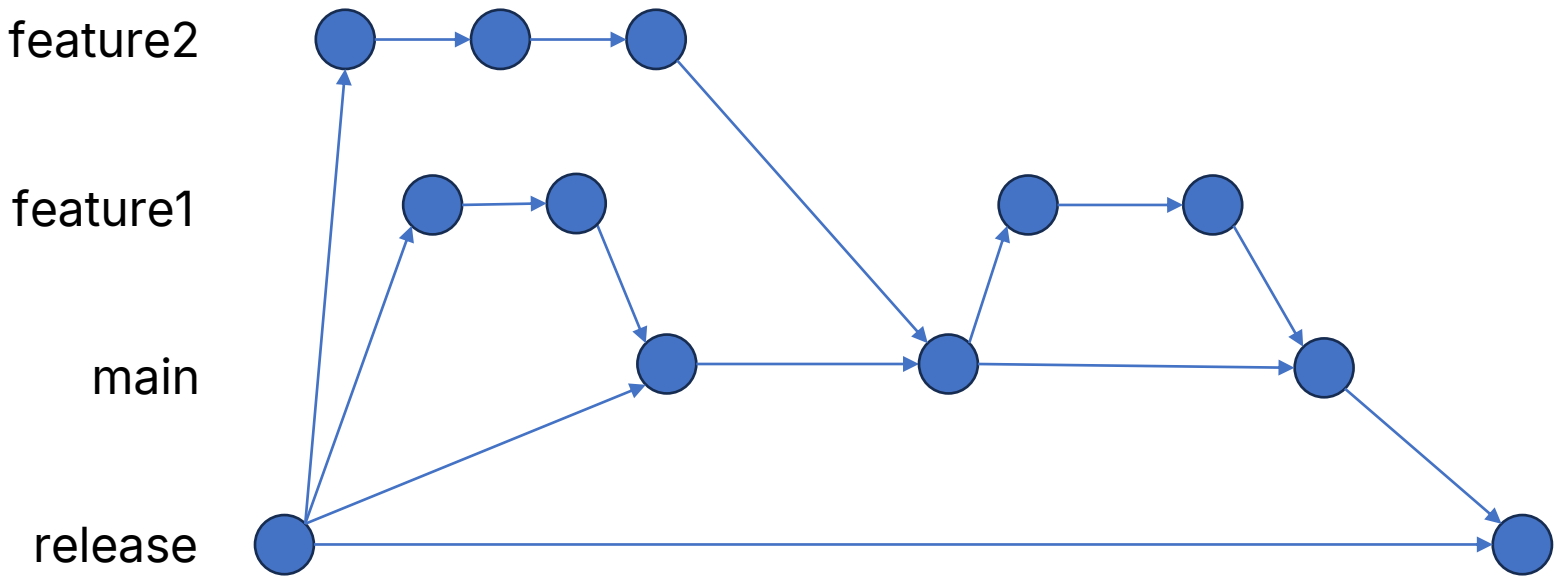
Branch workflows

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



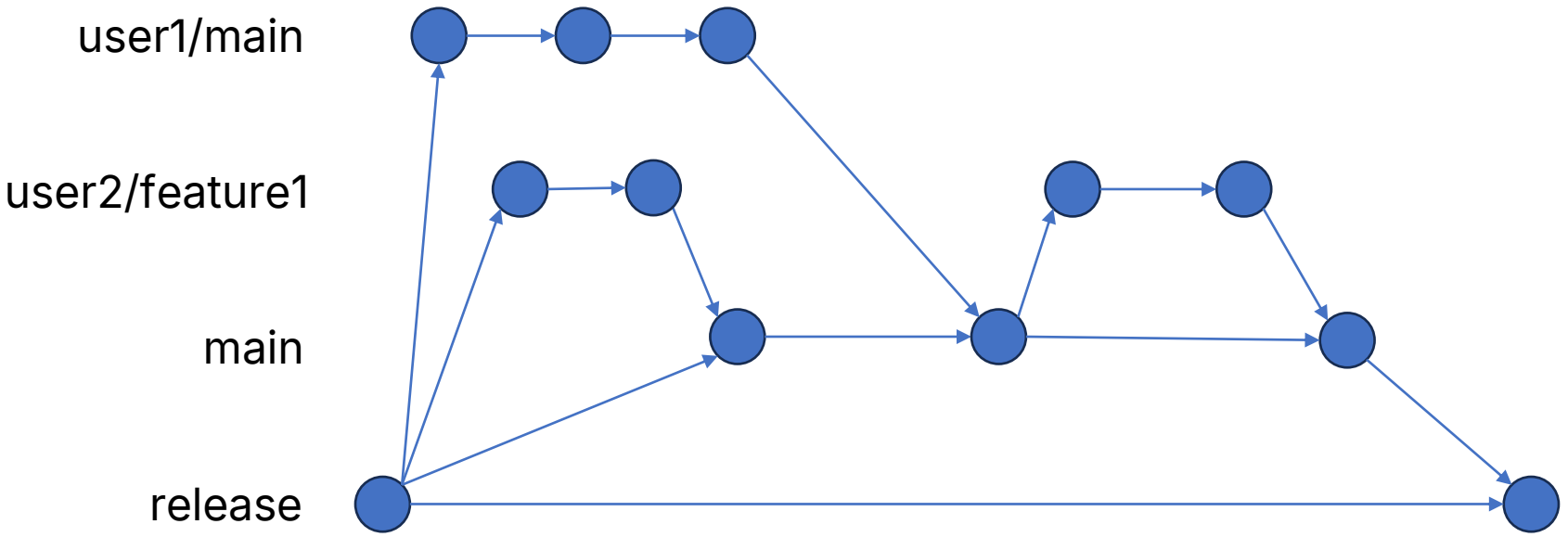
Branch workflows

- 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



Branch workflows

- Branches can be user forks too



Questions?

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
  - Find them on GitHub: https://github.com/github/gitignore/blob/main/Python.gitignore

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