Gitflow

EECS 2311 - Software Development Project

Click to edit Master text styles
Second level
Third level
Fifth level

Wednesday, January 22, 2020

YORK UNIVERSITÉ UNIVERSITY

Git: Fast-forward merges

- You pull the latest code from your group's online repository
- You work for a while making local commits
- When you push, you get an error: Not a fast-forward merge
- What has happened?
- How to resolve this problem?

What has happened?

- A teammate has pushed changes to the online repository while you were making your own changes
- These changes are conflicting with yours
- Must resolve the conflict

How to resolve this

- Pull from the online repository first
  - Resolve any conflicts if necessary
- Then, push again
- For a visual explanation, see: https://www.campingcoder.com/2018/03/git-merges-demystified/
  - Link is on course website as well

Gitflow

- Git provides the ability to create and switch between branches
- Unless there is some sort of workflow that determines what each branch is for, things can get messy pretty fast
- Gitflow is a convention for branch naming that we'll use in this course

Gitflow branches

- **Master**: Reflects a production-ready state
  - For us, system versions that can be demoed
- **Develop**: Reflects a state with the latest delivered development changes for the next release
  - When the develop branch is stable, it is merged into the master branch
- **Feature** branches
- **Hotfix** branches
- **Release** branches

Feature branches

- May branch off from: **develop**
- Must merge back into: **develop**
- Used to develop new features
- Branch naming convention: anything except master, develop, release-*, or hotfix-*

Feature branches in EECS 2311

- Each student **must** have their own development branch
- The branch name **must** contain your name
- Feature branches are often not pushed online, but in this course they **must**
- Your participation in the group submission will be based on the commits on the branches in your github repository

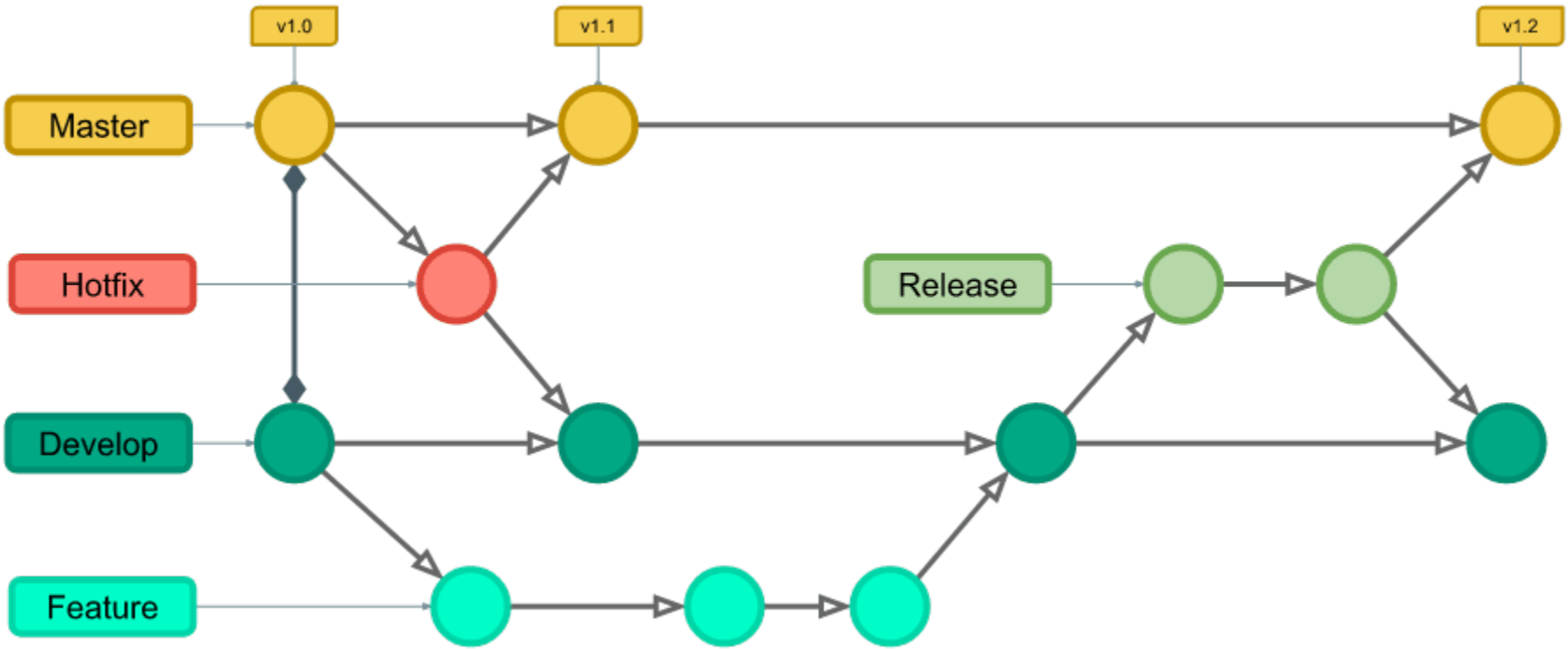
Release branches

- May branch off from: **develop**
- Must merge back into: **develop** and **master**
- Branch naming convention: **release-***
- Created just before a major release to ensure the release is production-ready
- Once the release branch is created, then the develop branch can receive features being developed for future releases

Hotfix branches

- May branch off from: **master**
- Must merge back into: **develop** and **master**
- Branch naming convention: **hotfix-***
- Created when a critical bug is discovered in a production release
- You may use release and hotfix branches in this course, but they are not required

Gitflow example



https://www.campingcoder.com/2018/04/how-to-use-git-flow/

Requirements Document

- Describes ***what*** the system does for its client/ customer, not how it does it
- Contains **use cases** for the system
- Contains **acceptance test cases**
- No particular format required for this course
- First draft due on Sunday (group submission)

Lab Task

- Each team must create and start using the master and develop branches
- Demonstrate to the TA switching to the master branch and running the app
- Each member of the team must create their own branch and demonstrate merging something to the develop branch
- All individual branches must appear on github