More on Git Documentation

EECS 2311 - Software Development Project

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

Tuesday, January 29, 2019

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
- Your project must have at least a **master** and a **develop** branch

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**
- Branch naming convention: anything except master, develop, release-*, or hotfix-*
- Used to develop new features
- Typically exist only in the developer repositories, not in the online one

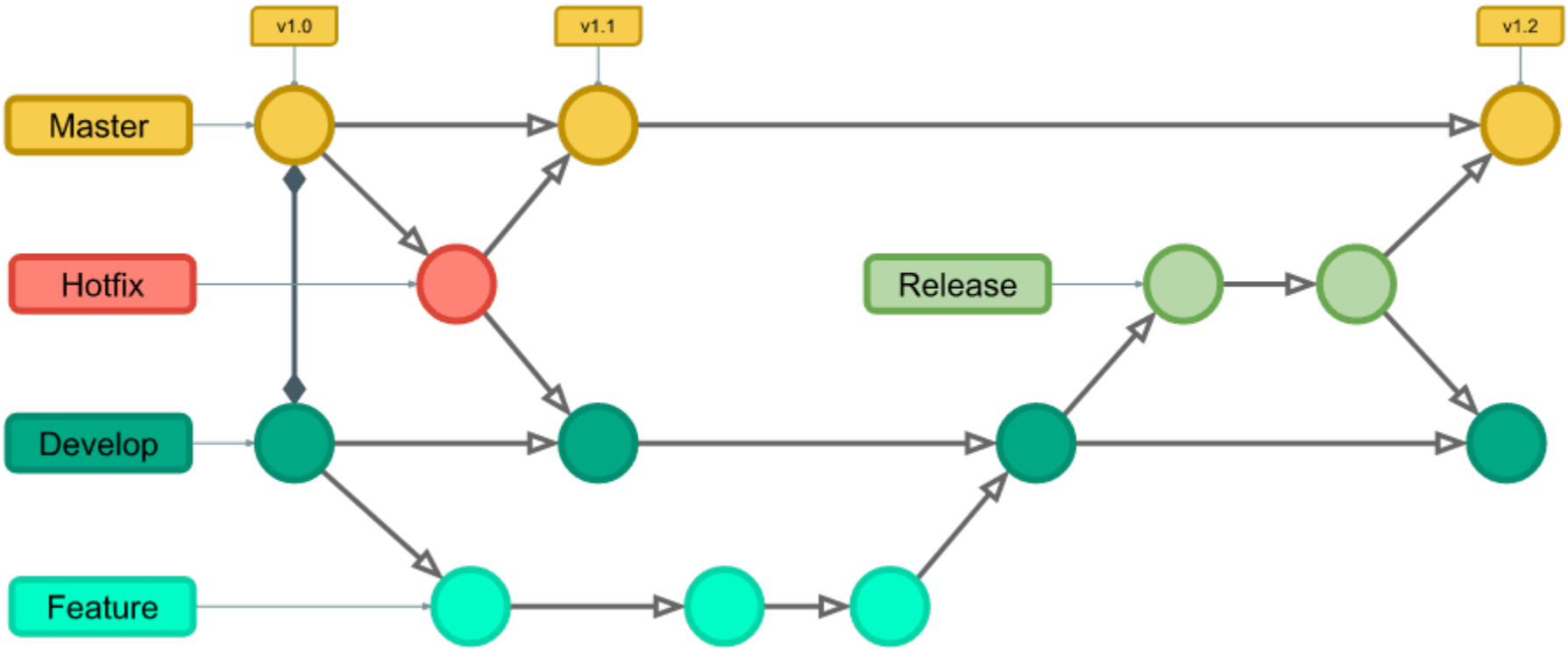
Release branches

- 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

Gitflow example



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

Midterm submission details

- Rubric posted on the course website
- Four grade components

- Requirements document
- Testing document
- User Manual
- Implementation

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

Testing Document

- Describes the test cases that have been implemented
- Includes discussion on how these test cases were derived
- Includes discussion on why these test cases are sufficient
- Includes test coverage discussion (next week's topic)
- No particular format required for this course

User Manual

- Describes how to install the system on the client's site
- Describes major use cases
- Contains screenshots or even links to video
- Must be easy to understand by someone who is unfamiliar with the system

Lab Task

- Each member of the team must create a first draft of one of the required documents
- All documents must be committed to your github repository both as source (Word, latex, Pages etc) as well as PDF
  - Create a folder called Documentation
- Present your document to your TA for initial feedback
- Use github to share updates to these documents throughout your project