CMPT 276 Class 08: Git Branches and Workflows

Dr. Jack Thomas
Simon Fraser University
Summer 2020
Today’s Topics

• Two more advanced Git features:

  – Using **Issues** to track features and bugs.

  – Using **Branches** to work on those features and bugs.
Issues In GitLab

• GitLab tracks **Issues:**
  – Bug reports and feature requests
  – Demo: Show issues in an active repo

• Value of Issues
  – Use as product’s **backlog**
  – Assign issues to a dev to show who's working on it
  – Update issues with extra info as needed
Branches

- **Master**: Main source code branch in a Git repo.
- **Head**: Latest code on master.
- Too chaotic to have many teammates constantly committing code to master.
  - **Solution**: Create feature branches

**Branch**

- Do work on a separate track (the branch) from the Master
- Commit changes to your branch
- When the feature is ready, merge the branch back to the Master
Issue and Branching Overview

**GL** = done in GitLab

**AS** = done in Android Studio

- **GL**: Pick an issue to implement & create branch.
- **AS**: Checkout branch, make changes, commit & push changes to the branch.

When feature is ready:

- **AS**: Merge Master to Feature branch (resolving conflicts); commit/push changes.
- **GL**: Create merge request to merge branch to Master.
- **GL**: Branch is deleted when merge request accepted. (manually remove merged local branch)
Issues and Branching

1. **Create an issue** for a bug or feature
   - Implementing a feature or fixing a bug should start with a GitLab issue.
   - Ex: Issue 14: "Add help button to game activity"

2. **Assign the issue** to yourself
3. **Create a feature branch** in GitLab

   - GitLab names the branch to start with the issue number.
   - Ex: 14-game-help-button
   - In Android Studio:
     a) **Fetch** to get new branch names: VCS -> Git -> Fetch
     b) **Checkout** the branch: Bottom-right “Git” button. Under remote branches, select your new one. On sub-menu, select checkout
   - Your work goes into the branch, not the master.
Issues and Branching

4. **Work** on your branch
   - Do your work changing files
   - Check-in your changes via Git:
     - Add: changes ready to be committed
     - Commit: put changes into local repo on branch
     - Push: push to remote repo on branch
Issues and Branching

5. **Merge** Master to Feature Branch
   – Get latest from master’s HEAD
   – In Android Studio: VCS --> Git --> Merge Changes...
   – Resolve merge conflicts; test, add/commit/push any changes.

6. Submit a **Merge Request** via GitLab
   – Create request to merge your branch back to master
   – Since you already merged Master to Feature Branch, there should be no conflicts.
   – GitLab will close issues associate with merge request; Otherwise, have message include “Fix #14”
Managing Merge Requests

• Team members see merge requests
  – **Code review**: Comment on problems they see in the code (possibly leading to new commits to fix)
  – Thumbs-up/down for voting

• **Repo Manager** accepts the merge request
  – Accepting merge requests will:
    • Merge code to master (should be no conflicts)
    • Closes associated issue (if any)
    • Delete the source branch [optional; good practice to clean up]
**GitLab Workflow**

**Feature Branch, Merging Changes, Merge Request**

1. **Create GitLab issue.**
2. **Assign issue to self**
3. **Create feature branch**

**Sequence of Events**

- 1. **Pull to update.**
- 2. **Checkout local feature branch.**
- **Change files, commit, push.**
- **Change files, commit, push.**

**Teammate Feature Branches**

- **Master**
- **My Feature Branch**

**Legend**

- **In GitLab**
- **In Android Studio**

**Create & accept Merge Request**

- 1. **Pull.**
- 2. **Remove local feature branch.**
- 4. **Resolve conflicts.**
- 5. **Build and test.**
- 6. **Commit/push changes.**
- 7. **Switch to Master branch.**

**GitLab closes associated issue**
Recap – Merge Requested

• **Branches and Workflow**
  – Create GitLab issues.
  – Do work on a feature branch.
  – GitLab merge request to merge branch to master.