Git Branches and Workflow

Many features, many developers;
How can Git help?

Issues in GitLab

- GitLab tracks Issues:

  ..
  - Demo: Show issues in an active repo
- Value of Issues
  - Use as product’s backlog
  - Assign issue to a dev to show who's working on it
  - Update issue with extra info as needed

Branches

- .. Main source code *branch* in a Git repo.
- .. Latest code on master.
- Chaotic Commits
  - Too chaotic to have many teammates constantly committing code to master.
  - Solution:..
- Branch (Feature Branch)
  - Do work on separate track (the branch) from Master
  - Commit changes to your branch
  - When feature is ready,

    ..

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 issue for bug/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 issue to yourself
3. Create feature branch in GitLab
   - GitLab names the branch..
     - Ex: 14-game-help-button
   - In Android Studio, checkout the branch:

     ..

Issues and Branching (cont.)

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
5. Merge Master to Feature Branch
   - Get the latest code from master's HEAD:
     - In Android Studio: VCS --> Git --> Merge Changes...
   - Resolve any merge conflicts
   - Test, commit/push any changes.

Issues and Branching (cont.)

6. Submit a.. via GitLab
   - Create request to merge your branch back to master
   - Since you already merged Master to Feature Branch, there *should* be no conflicts.
   - If branch name starts with a number, GitLab will associate merge request with the issue. Otherwise, have message include “Fix #14”

Managing Merge Request

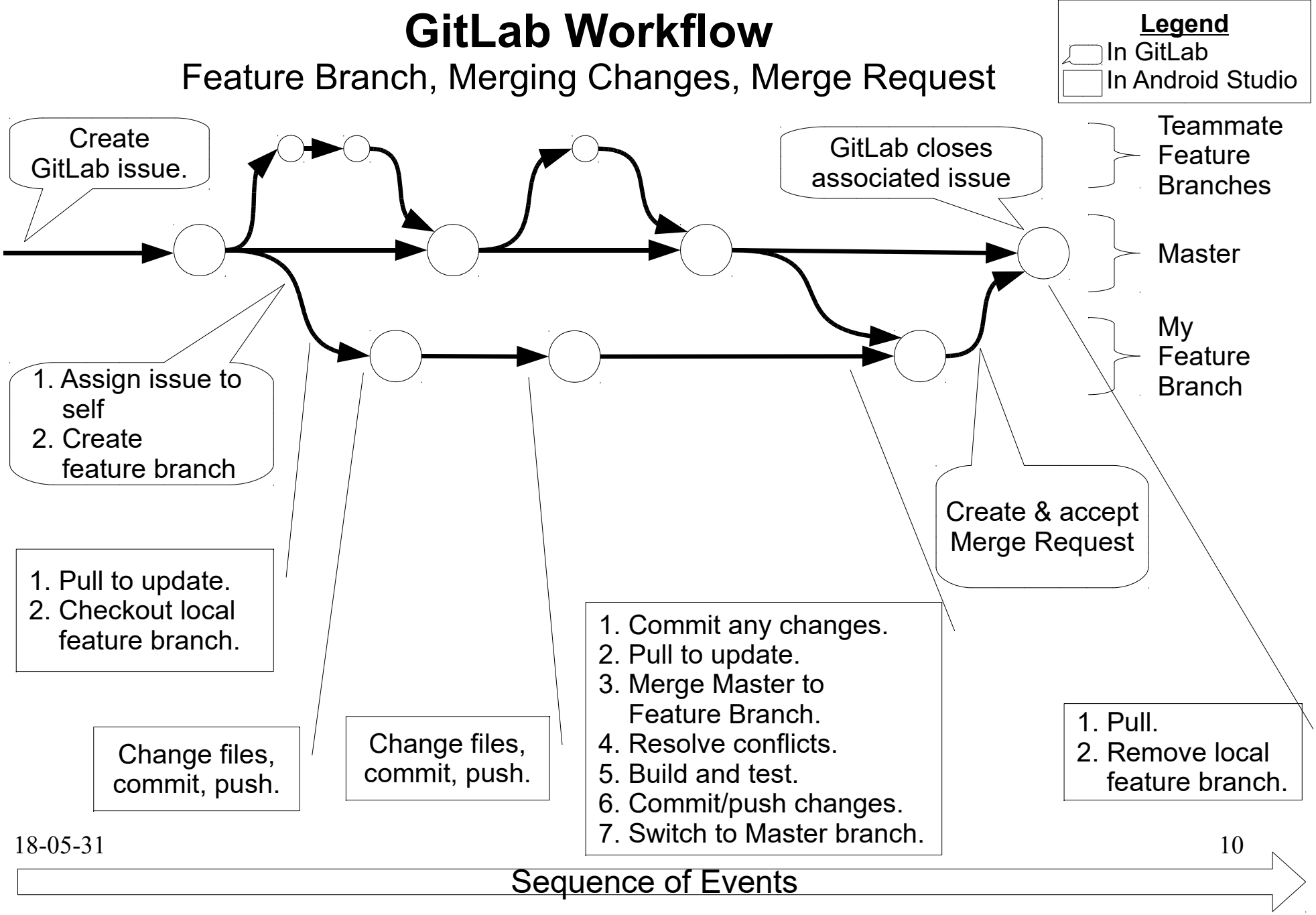
- Team members see merge requests and:
  - Code review:
    Comment on problems they see in the code
    (possibly leading to new commits to fix)
  - Thumbs-up/down for voting
- Repo Manager accepts merge request
  - Accepting merge requests will:
    - merge code to master (should be no conflicts)
    - ..
    - delete the source branch
      [optional; good practice to clean up]

GitLab Feature Branch Demo

- Setup
  - Create new Android Studio project
  - Create GitLab project; commit/push
- Create GitLab issue
  “Generate random number”
- Solve an issue
  - In GitLab: Assign to you & create feature branch
  - In A.S.: Checkout feature branch (bottom-right)
  - Code; Commit; Push
  - In A.S.: Merge master to feature (VCS-->Git-->Merge Changes...)
  - In GitLab: Create merge request
  - In GitLab: Accept merge request.
- Cleanup in Android Studio
  - Switch to master, pull, view log
  - Delete local branch (if needed)

GitLab Workflow

Feature Branch, Merging Changes, Merge Request



Review Exercise

What is the ordering of the following steps:

a) In GitLab: Assign to you & create feature branch

b) In A.S.: Merge master to feature branch (VCS-->Git-->Merge Changes...)

c) Create GitLab issue

d) Code; Commit; Push

e) In GitLab: Accept merge request.

f) In A.S.: Checkout feature branch (bottom-right)

g) In GitLab: Create merge request

Summary

- Branches and Workflow
  - Create GitLab issues.
  - Do work on a feature branch.
  - GitLab merge request to merge branch to master.
  - Git merge command