Git Branches and Workflow

Many features, many developers;
How can Git help?

Issues in GitLab

- GitLab tracks Issues:
  .. bug reports and feature requests

- 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

- ..Master: Main source code *branch* in a Git repo.
- ..Head: Latest code on master.
- Chaotic Commits
  - Too chaotic to have many teammates constantly committing code to master.
  - Solution:.. Create feature branches
- Branch (Feature Branch)
  - Do work on separate track (the branch) from Master
  - Commit changes to your branch
  - When feature is ready, ..merge branch back to 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 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.. to start with the issue number.
     - Ex: 14-game-help-button
   - In Android Studio, checkout the branch: .. your work goes into the branch not master.

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.. 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.
   - If branch name starts with a number, GitLab will associate merge request with the issue. Otherwise, have message include "Fix #14"

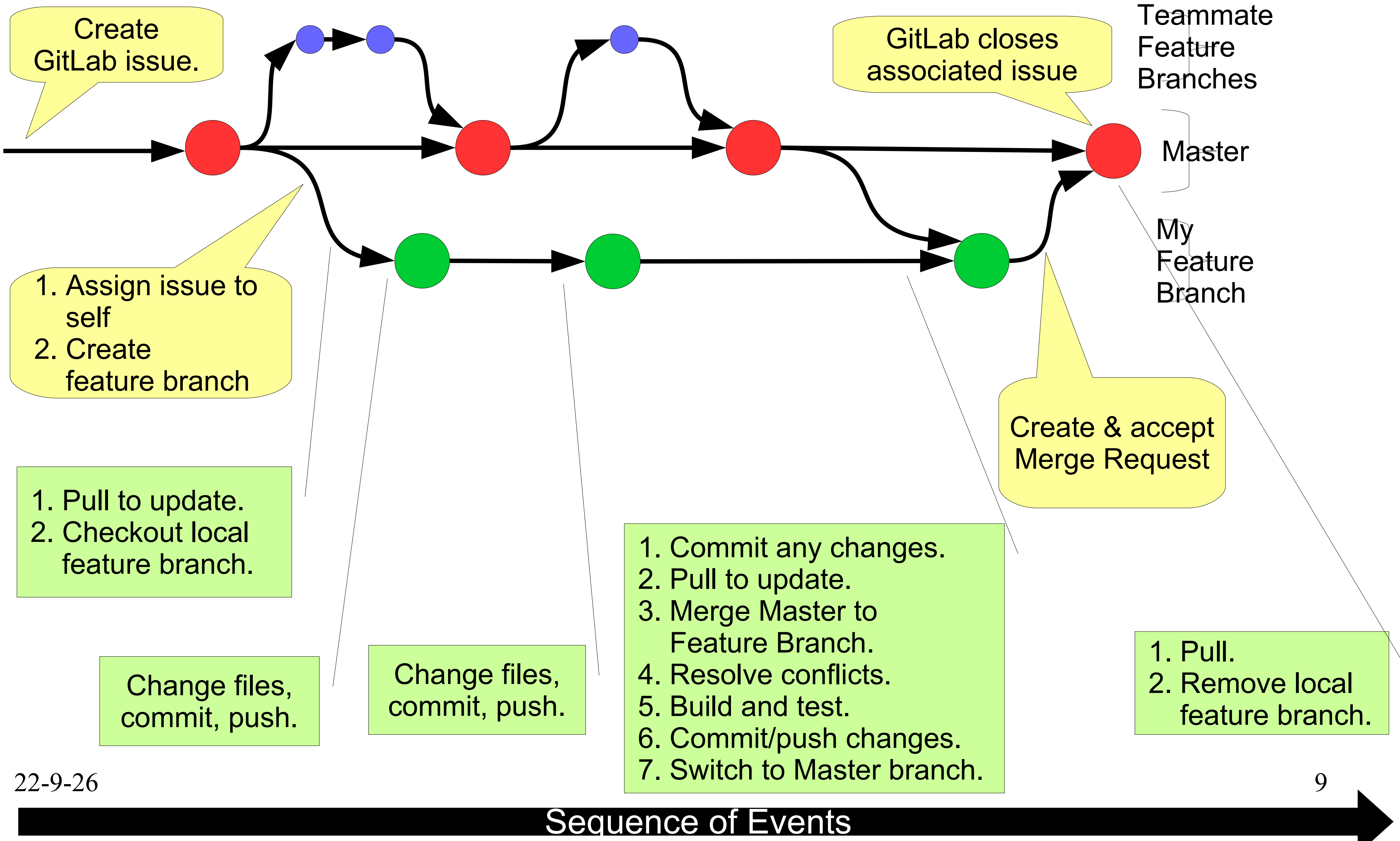
Managing Merge Request

- 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)
    - ..closes associated issue (if any)
    - delete the source branch
      [optional; good practice to clean up]

GitLab Workflow

Feature Branch, Merging Changes, Merge Request

Legend
🟨 In GitLab
🟩 In Android Studio



Create GitLab issue.

1. Assign issue to self
2. Create feature branch

1. Pull to update.
2. Checkout local feature branch.

Change files, commit, push.

Change files, commit, push.

1. Commit any changes.
2. Pull to update.
3. Merge Master to Feature Branch.
4. Resolve conflicts.
5. Build and test.
6. Commit/push changes.
7. Switch to Master branch.

Create & accept Merge Request

GitLab closes associated issue

1. Pull.
2. Remove local feature branch.

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