Git Review

Site Reliability Engineering

{mthree}

Overview

Learning Objectives

- In this module, we will do a quick overview of the core Git Commands

By the end of this module, you will be able to:

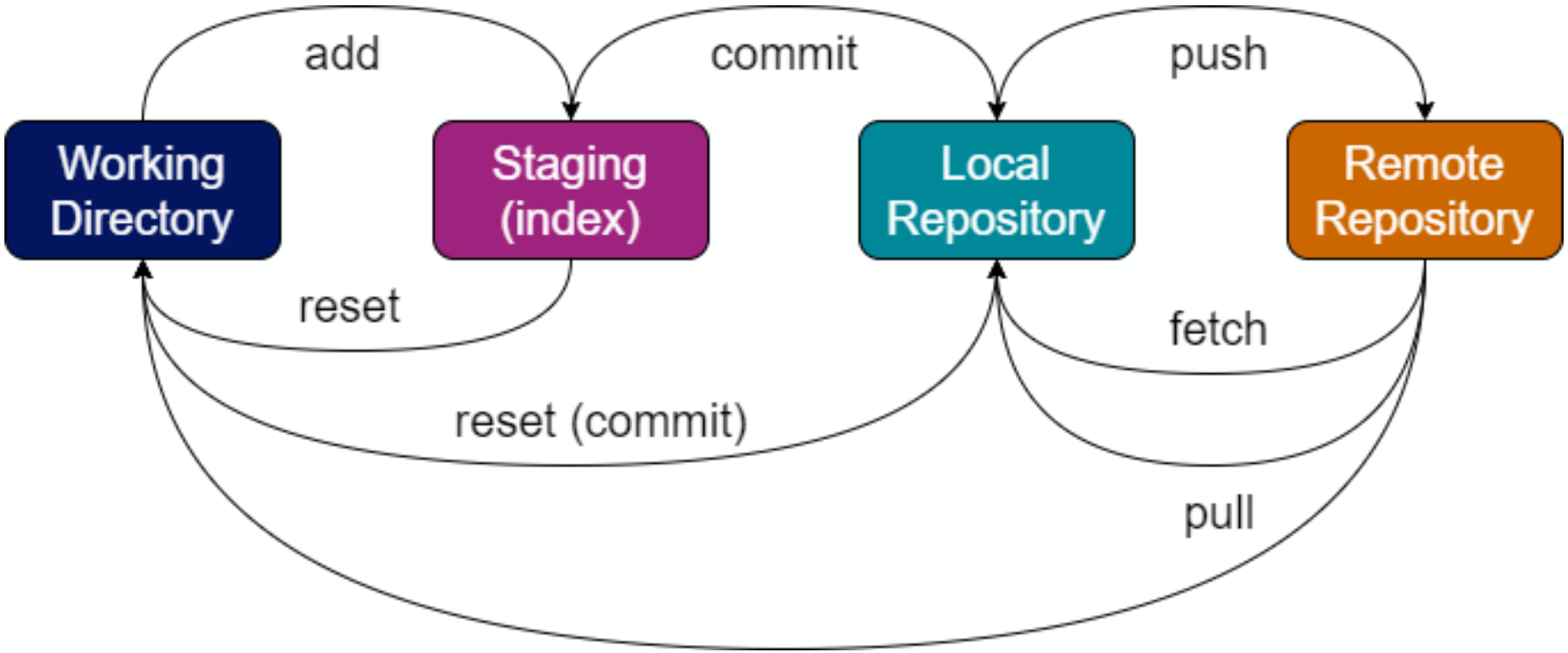
- >>> Use the Git commit cycle
- >>> Work with branching
- >>> Merge branches
- >>> Create and manage pull requests

{mthree}

What is Git?

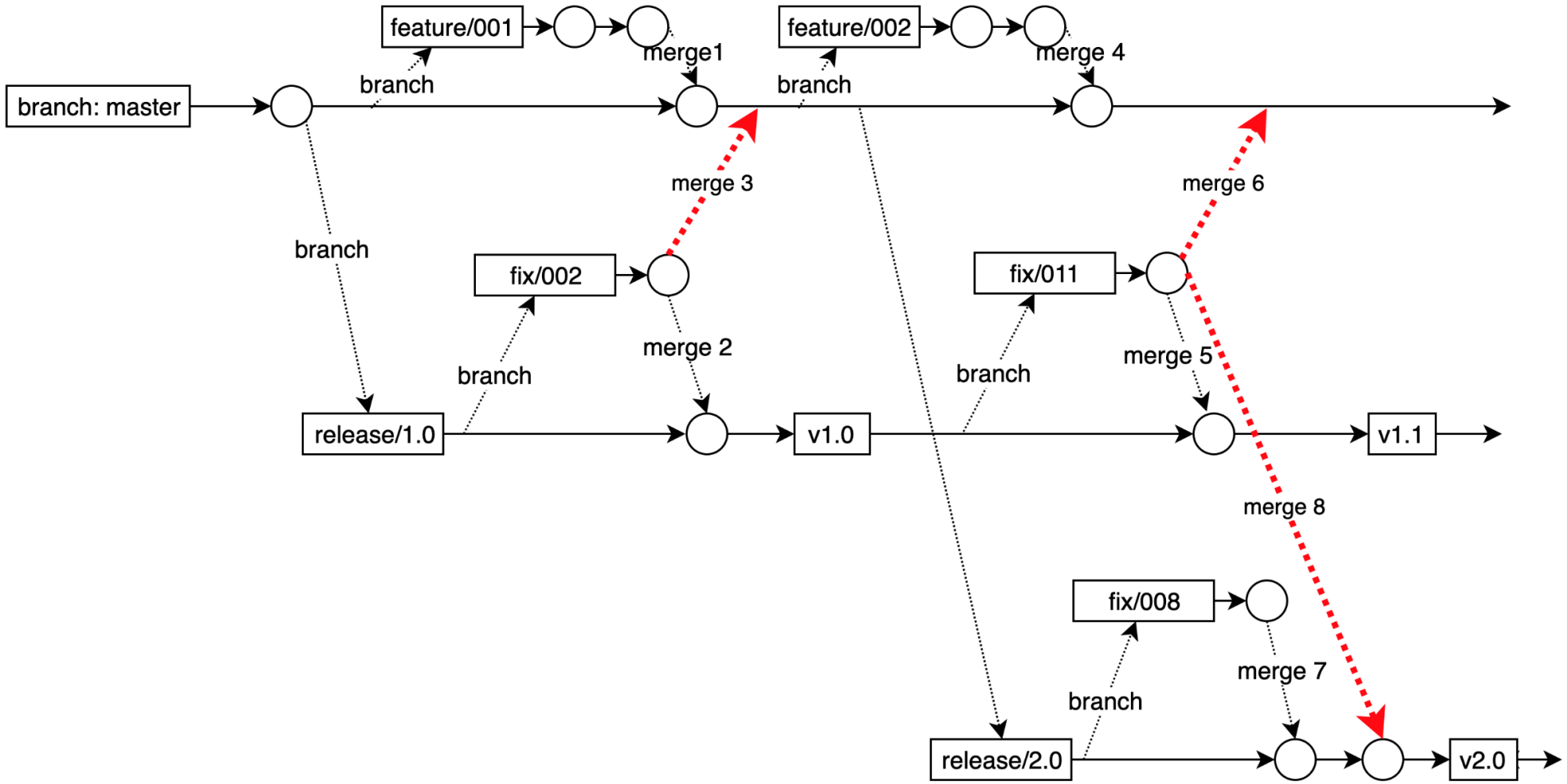
- Version Control Software (VCS)
  - >>> Tracks code versions independently of each other, giving developers more control over what code is in production
  - >>> Allows developers to easily revert code to a previous version
- Allows developers to share and maintain code as a team
  - >>> Each dev has their own copy of the code
  - >>> Code can be assigned to branches based on development status
  - >>> Branches can be merged into production code
- Combines the convenience of working on code locally with the advantage of sharing code online
- Widely used
- Extensive documentation
- Requires a local installation of Git
- Managed through a command-line interface such as Git Bash or Terminal

The Git Commit Cycle



{mthree}

Branching


feature/001
merge1
feature/002
merge 4
branch
branch
branch: master
merge 3
merge 6
branch
fix/002
fix/011
merge 2
merge 5
branch
branch
release/1.0
v1.0
v1.1
merge 8
fix/008
merge 7
branch
release/2.0
v2.0

Using a Git Repository – Step-by-Step

- Clone: Create a local copy of an existing repository
  - >>> Clone *https://github.com/The-Software-Guild/sre-git-refresh*
    - ~ `git clone https://username@github.com/The-Software-Guild/sre-git-refresh`
- Add files ready for history
  - >>> Create a new directory in the format of *courseCode-firstname*
    - ~ e.g., `mkdir ?XXX-steve` (? = r – reskill c – cohort)
  - >>> Change into this new directory
    - ~ `cd ?XXX-steve`
  - >>> Create a file named `myfile.txt` in this directory
    - ~ Use your favourite text editor on your system to do that
      - On Windows, `notepad myfile.txt`
      - On Linux or Mac, `nano myfile.txt`
  - >>> Stash your changes ready to add to the history
    - ~ `git add --all` (two dashes in front of *all*)
    - ~ This will add all new, modified, or deleted files/directories

{mthree}

Pushing: Step-by-Step

- Make your code available in GitHub
  - >>> Commit your change to create a history point
    - ~ `git commit -m “My first commit”`
    - ~ You may need set your name and email with `git config --global user.name “Firstname Lastname”`
  - >>> Push your changes to the central repository
    - ~ `git push`

- Conflict or error on push?

{mthree}

Unable to Push?

- Dealing with the error
  - >>> The first person has changed the central history
- We need to get the latest updates: git pull
- Now try to push again: git push
- Before you commence work on a local copy of a shared repository: git pull
  - >>> This ensures that your local files include any changes pushed to the repo
  - >>> **PULL OFTEN**

{mthree}

Branching Step-by-Step

- Create a branch in the sre-git-refresh repository on your git command line
  - >>> The name of the branch should be *class-firstname*
  - >>> `git checkout -b ?XXX-steve` (? = r – reskill c – cohort)
- Go to your directory and add another file to that directory named *mybranch.txt*
  - ~ Edit the file to describe your favourite meal
  - ~ Save the changes to the file
- Now we need to add and commit the changes as you have done previously
- Now push to the central repository, but we need to name our branch
  - >>> `git push –u origin firstnamelastname`
- Now add your favourite drink to the mybranch.txt file
  - >>> Save, add, and commit
  - >>> This time use just git push after you have run the add and commit

{mthree}

Pull Requests

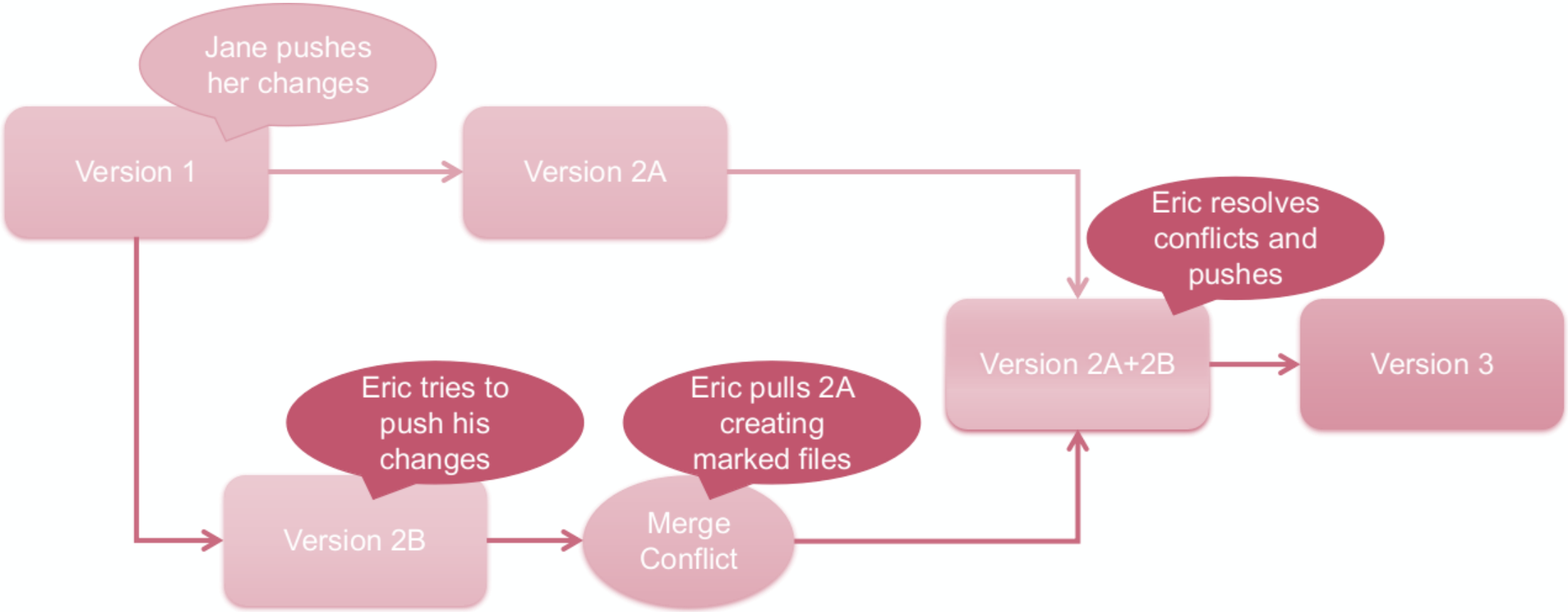
- Team members review your work and approve or decline
  - >>> Like a peer review
  - >>> Declined PR will require new submission
- Once approved, your branch can be merged
  - >>> Merge happens through web console
- All conflicts must be resolved before the merge is allowed
  - >>> PR does not need to be recreated
- Request that your work be added to the release branch

{mthree}

Pull Request: Step-by-Step

- Create a pull request for your branch to be included in main
- The instructor will pick a pull request and merge it
- Check your pull request and see if it can be merged
  >>> If the merge button is grey, then it cannot be merged
- To fix the issue
  >>> We need to update the branch to match main (or master)

Merge Conflicts



{mthree}

Step-by-Step: Resolve a Merge

- Back to your git window
- Check which branch we are on: `git branch`
- If we're on our branch, we need to switch to the main branch: `git checkout main`
- Update our local copy of main: `git pull`
- Switch back to our branch: `git checkout ?XXX-steve`
- Update our branch to include the updates in main: `git merge main`
- Now we can push our branch (no need to add or commit): `git push`
- Then check the Pull Request to see if you can merge
  >>> If not, go back through this process from checking out main and updating

{mthree}

Summary
Q & A

{mthree}

References

- Stack Exchange: Banjara. (15 March 2019). Git branches: Merging issues while having multiple release branches. Retrieved from https://softwareengineering.stackexchange.com/questions/388681/git-branches-merging-issues-while-having-multiple-release-branches

{mthree}