Embedded Software Essentials

Version Control

C1 M1 V9

Be Boulder.

University of Colorado Boulder

Version Control Systems (VCS)

- Software package that allows users to track changes

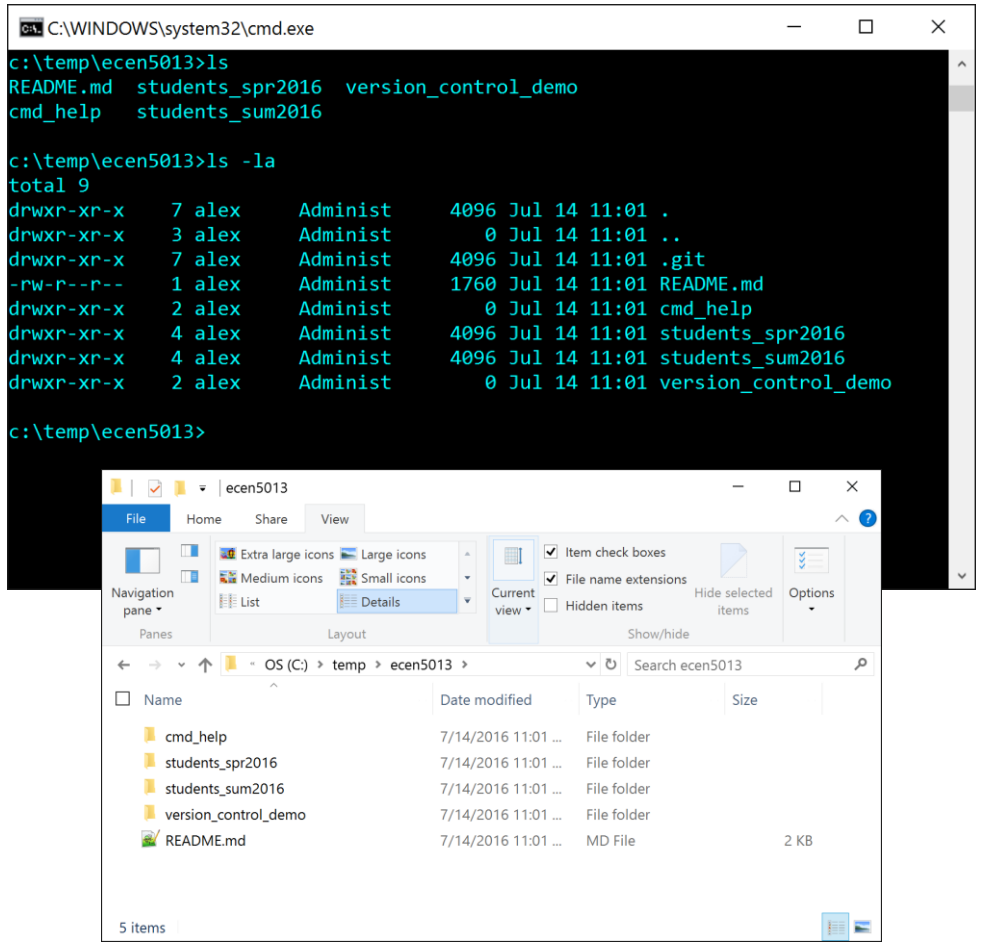
- Many different types
  - Concurrent Versions System (CVS)
  - Subversion (SVN)
  - Mercurial
  - IBM Rational ClearCase
  - **Git**

Used by:
- Linux Kernel
- Google's Android
- Twitter
- GNU Toolchain

VCS Features

- Allows you to track:
  - Software
  - Documents
  - Build information
  - Software configuration information

- **Repository**: Collection of tracked files
  - Acts like a normal file system



Windows Perspective

Be Boulder.

University of Colorado Boulder

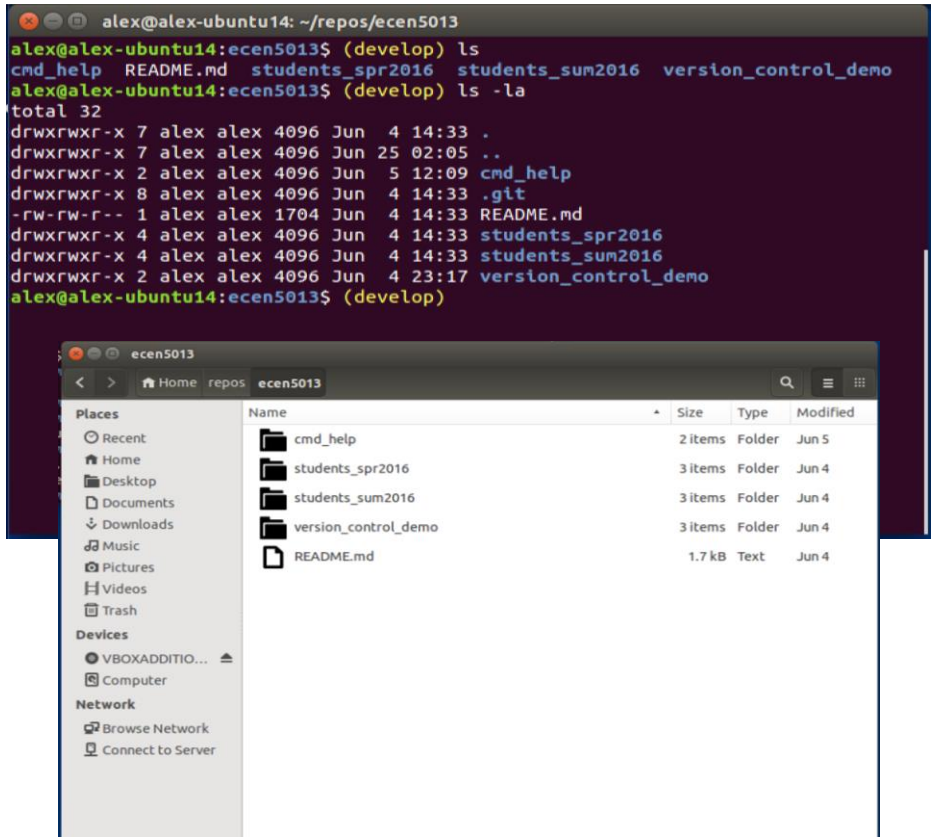
VCS Features

- Allows you to track:
  - Software
  - Documents
  - Build information
  - Software configuration information

- **Repository**: Collection of tracked files
  - Acts like a normal file system

- Has extra hidden configuration files
  (DO NOT EDIT THIS FOLDER)

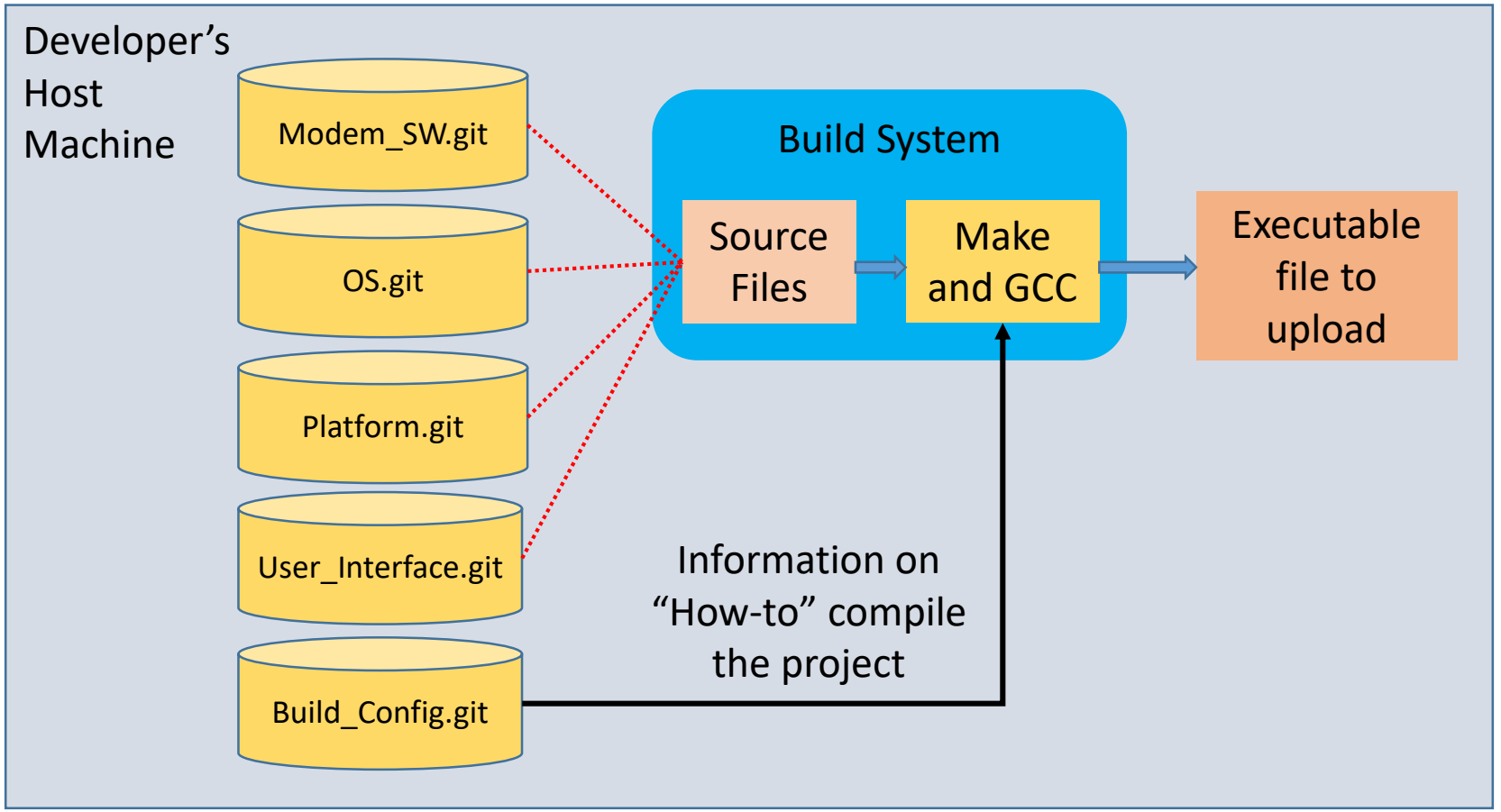
→ **.git folder**



Linux Perspective

Multiple Repositories (Repos)

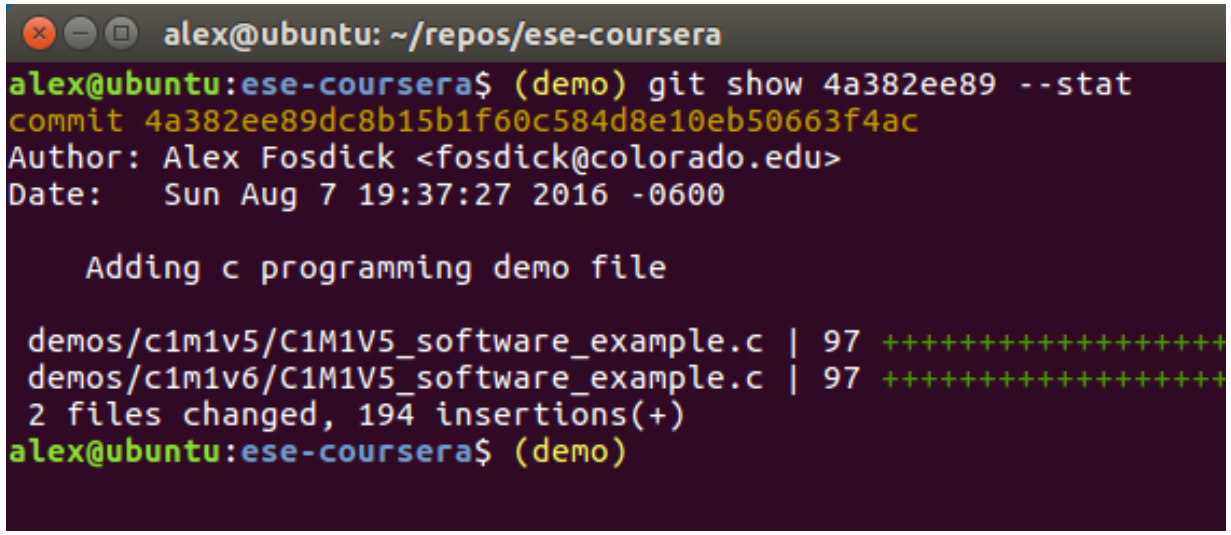
- You can have many repositories contribute to a single software product
- Each individually managed in a separate git repo



Example Project for an Android phone

Commits

- Tracked changes, in logical groupings
  - Single/Multiple file changes
  - New or Removed files
- Commits have **metadata**
  - Extra information with regards to tracking the changes made
- Metadata includes info like
  - Author of Change-set
  - Data of Change
  - File Change List
  - Commit Comment
  - SHA-1 Commit ID
  - Parent commit



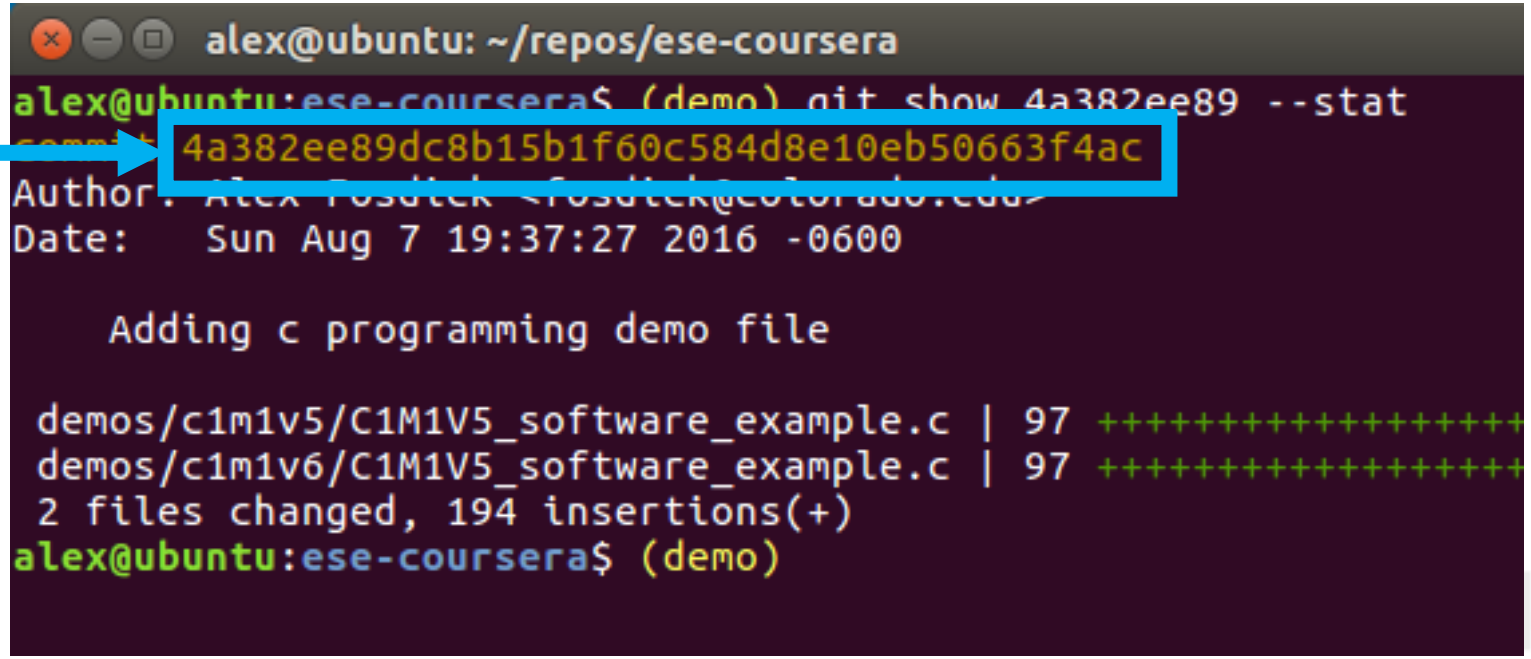
Be Boulder.

University of Colorado Boulder

Commits

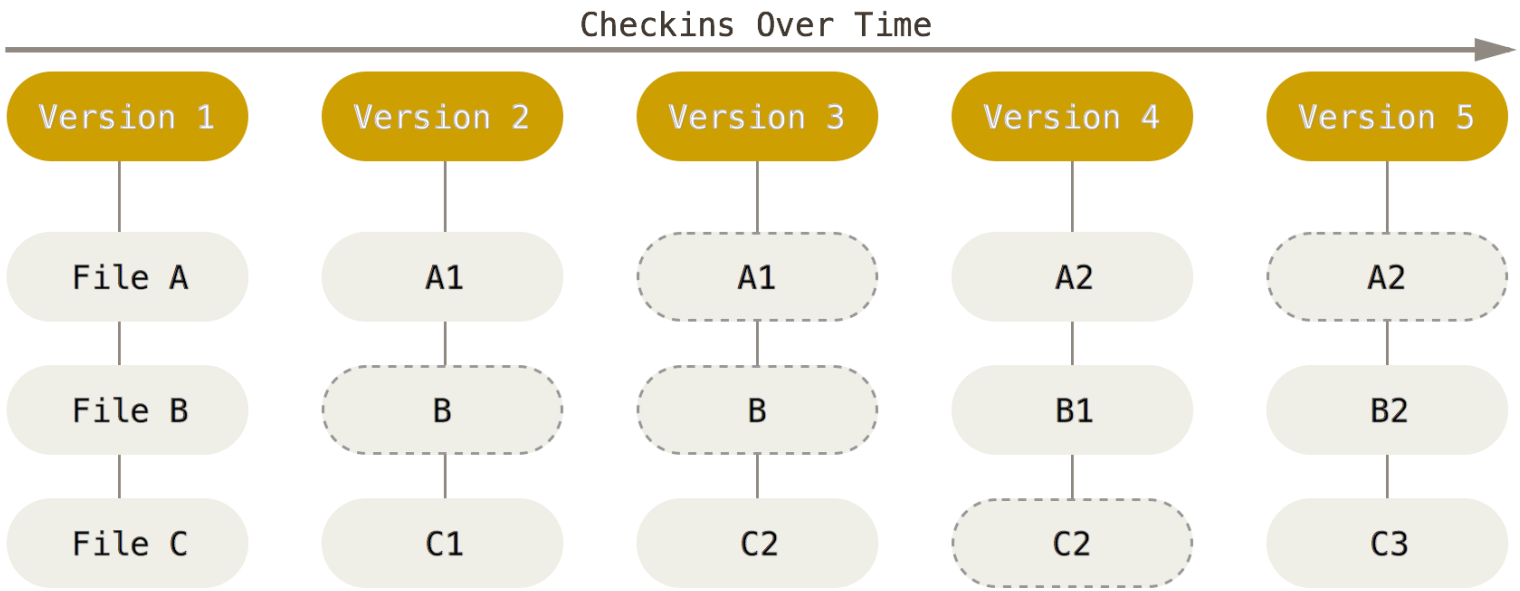
- Commit SHA - A 40 Character Unique ID
  - Reference point for your change set

SHA-1 Commit ID



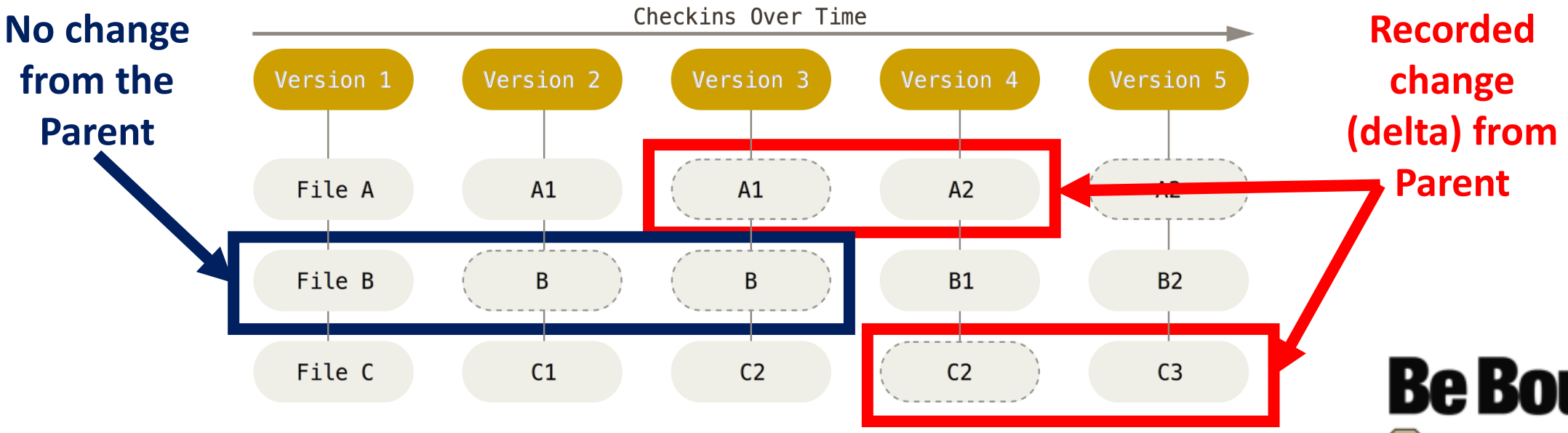
Commits

- Commits (Checkins) contain info about the **Parent commit**
  - The commit before the current changes



Commits

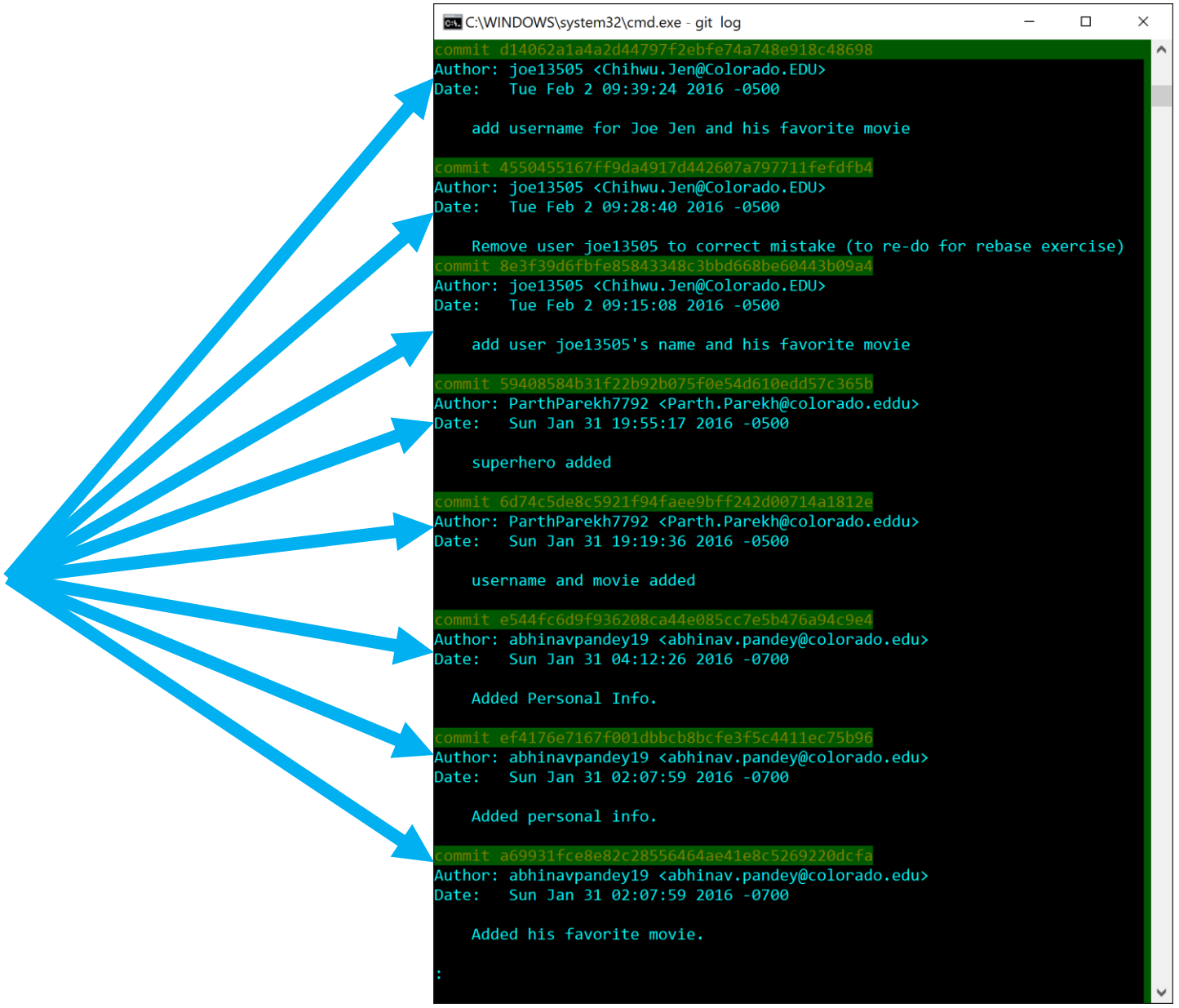
- Commits track the “changes” from the current commit and the **Parent commit**
  - Not all files have changes from the parent



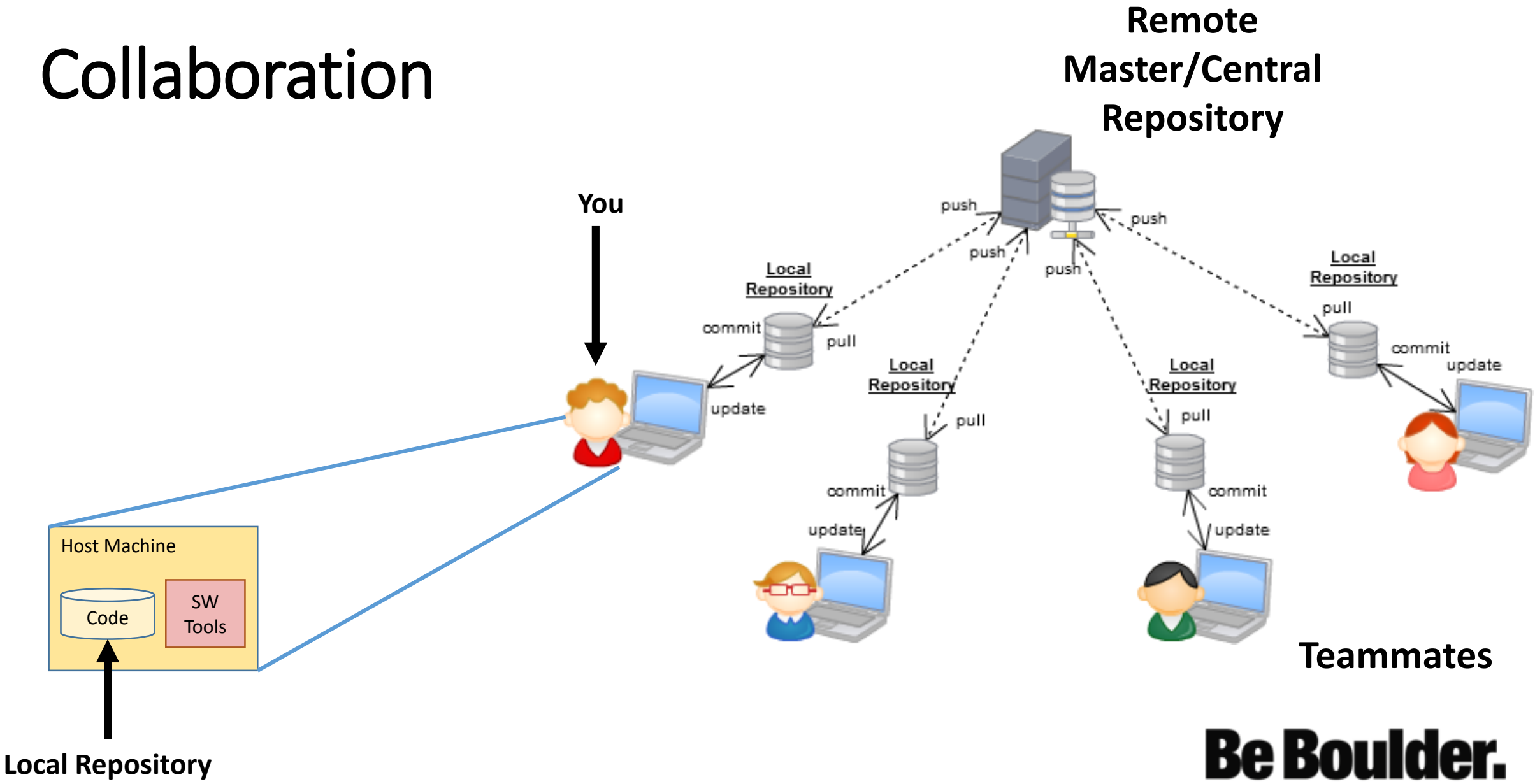
Git Log

```
$ git log
```

Different Commits with a list of some of the commit metadata information



Collaboration

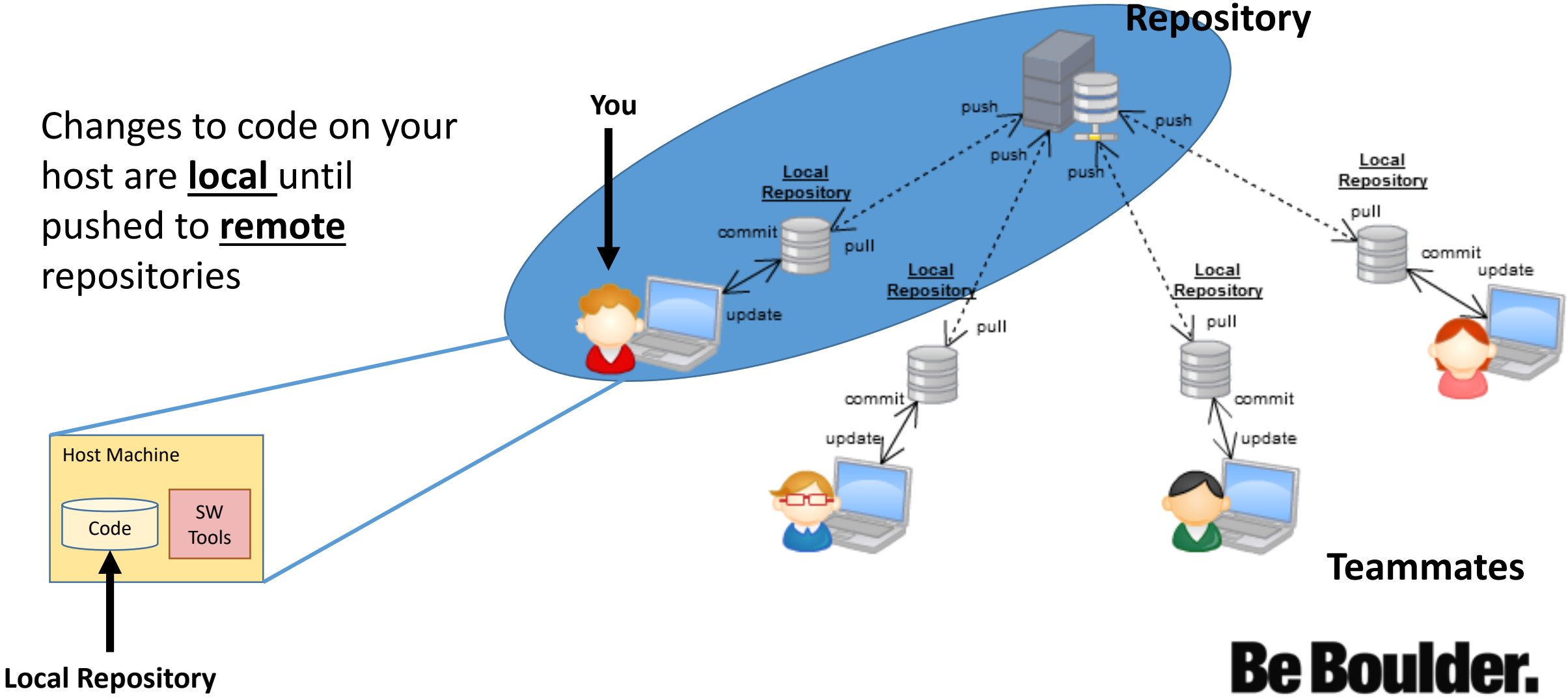


Be Boulder.

University of Colorado Boulder

Collaboration

Changes to code on your host are **local** until pushed to **remote** repositories



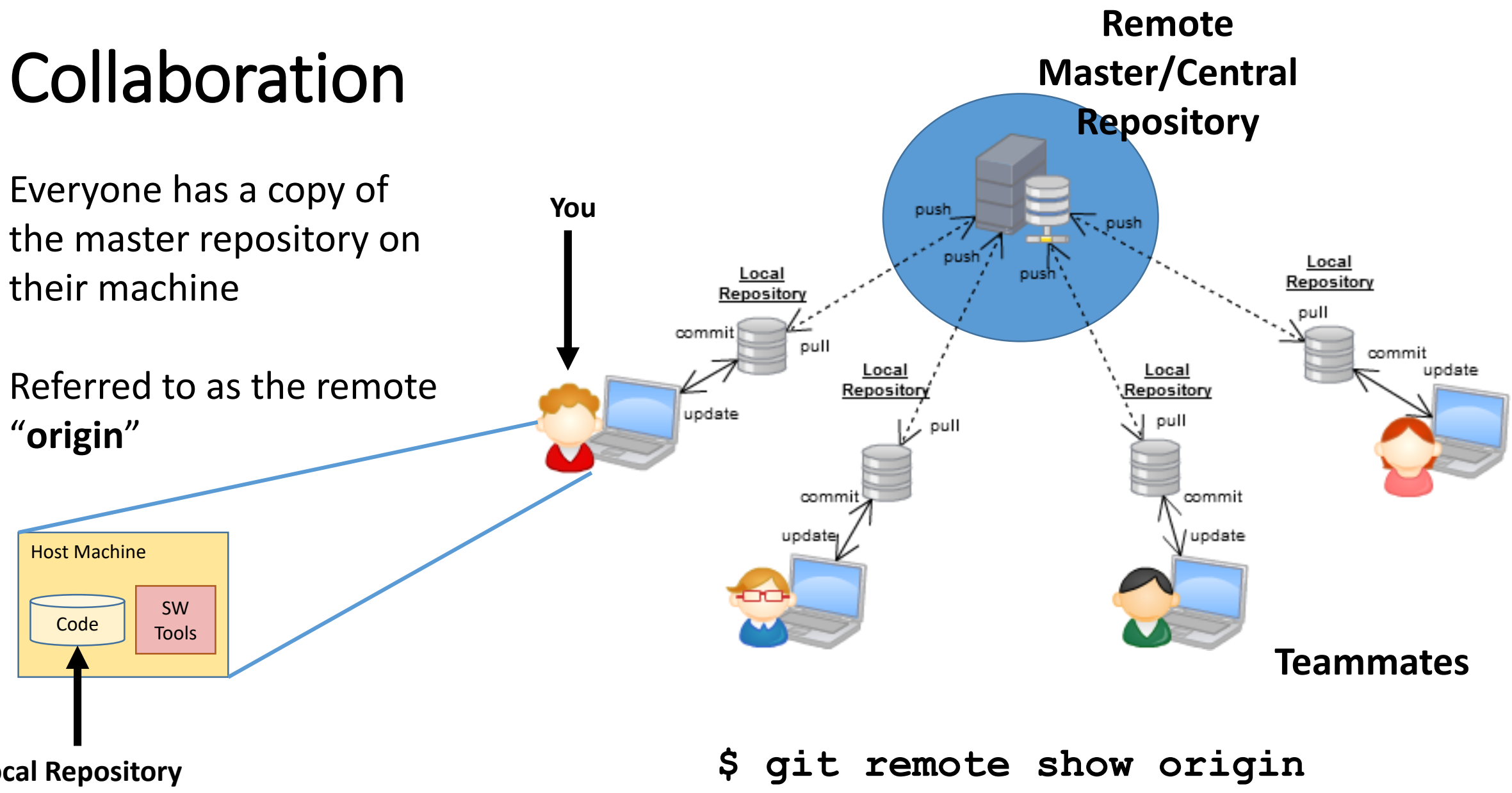
Be Boulder.

University of Colorado Boulder

Collaboration

Everyone has a copy of the master repository on their machine

Referred to as the remote "**origin**"

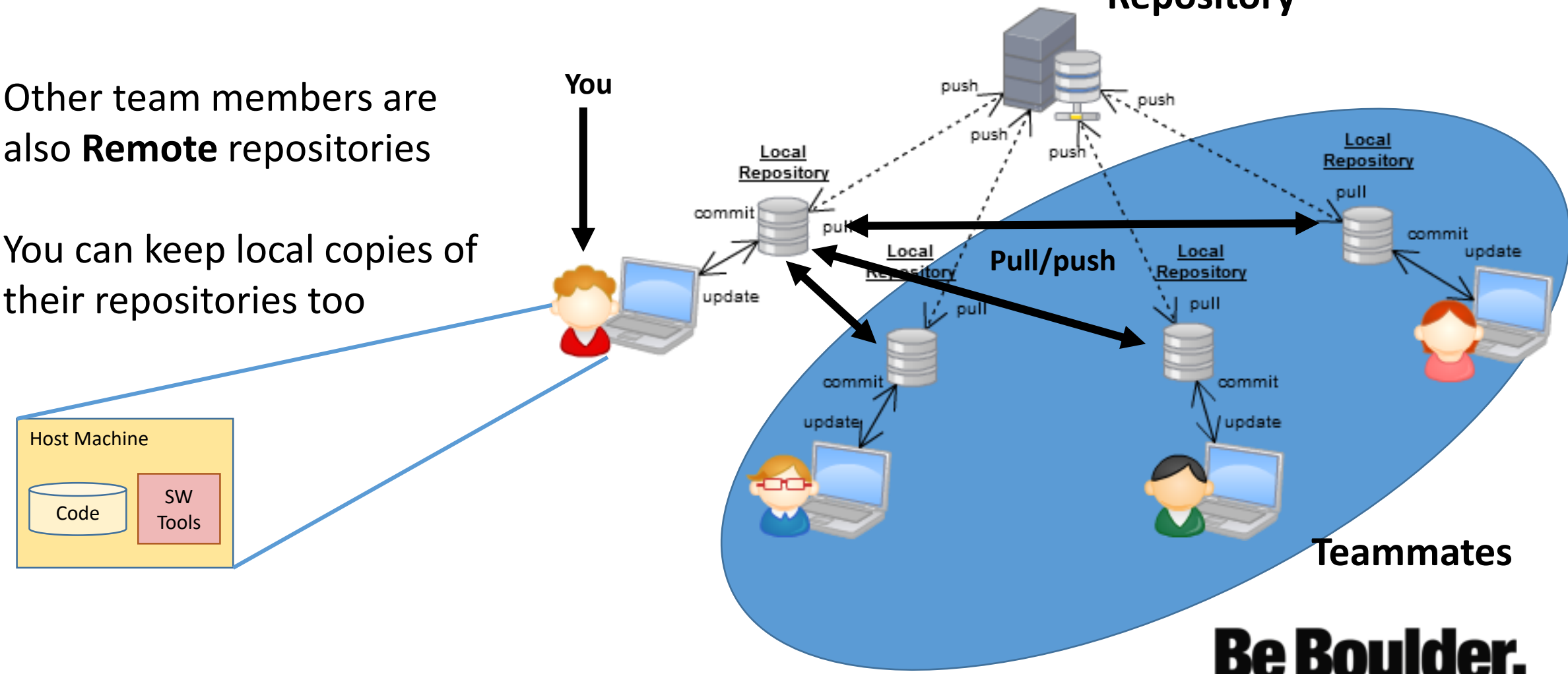



```
$ git remote show origin
```

Collaboration

Other team members are also **Remote** repositories

You can keep local copies of their repositories too



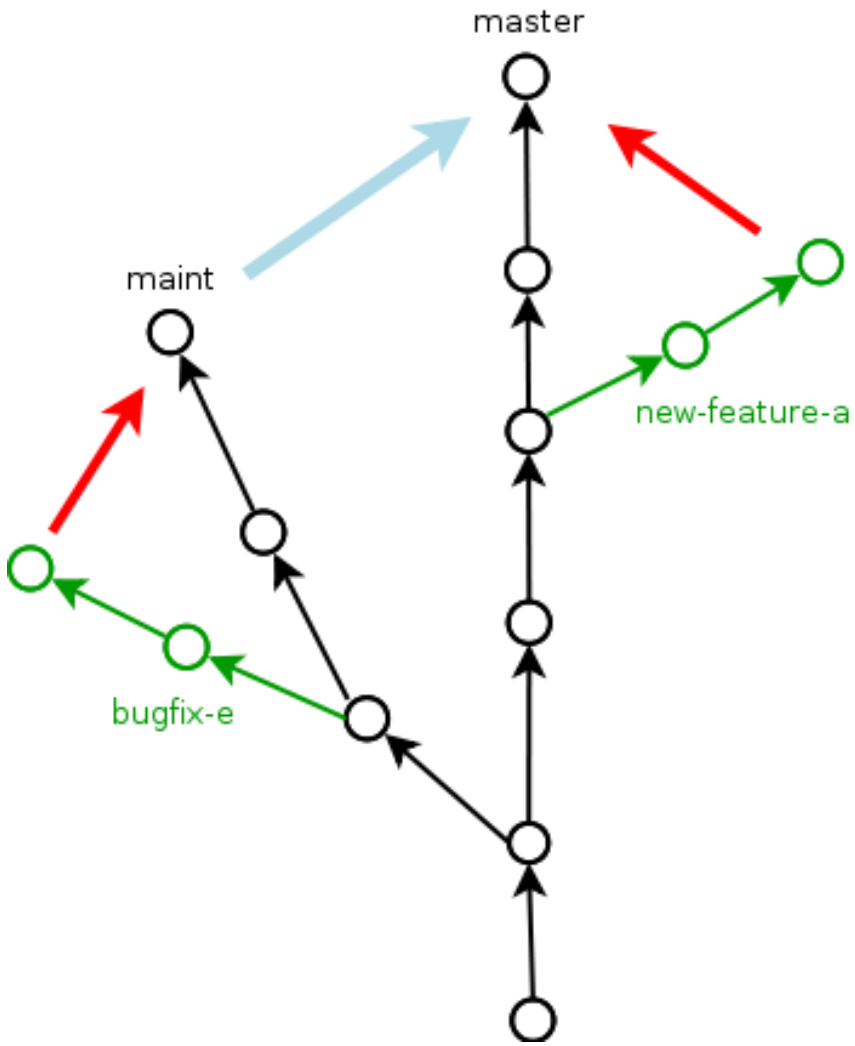
Be Boulder.

University of Colorado Boulder

Workflow

1. Create a **development branch**

Branch: Helps organizes a change set from others



Workflow

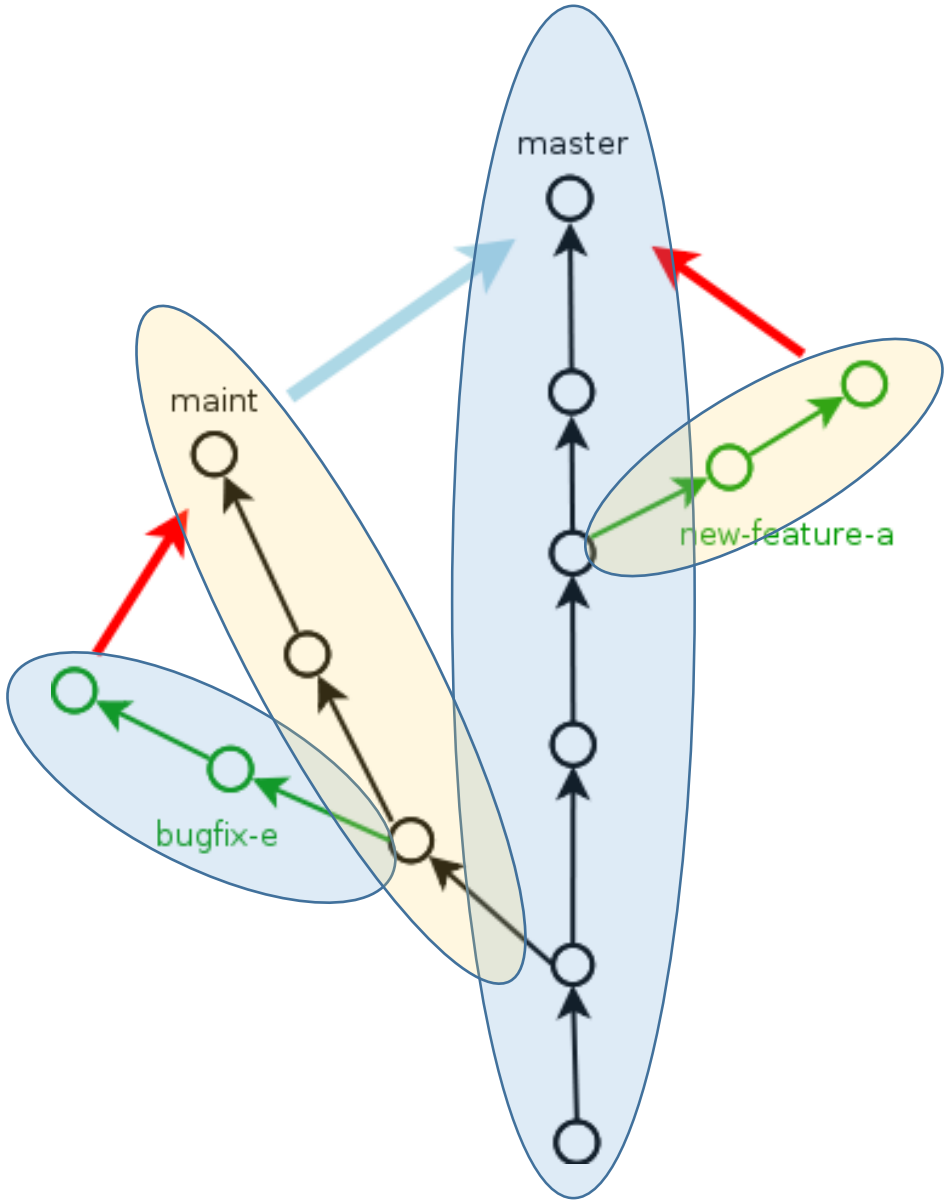
1. Create a **development branch**

Branch: Helps organizes a change set from others

Four Branches each with many commits

- master, maint, new-feature-a, bugfix-e

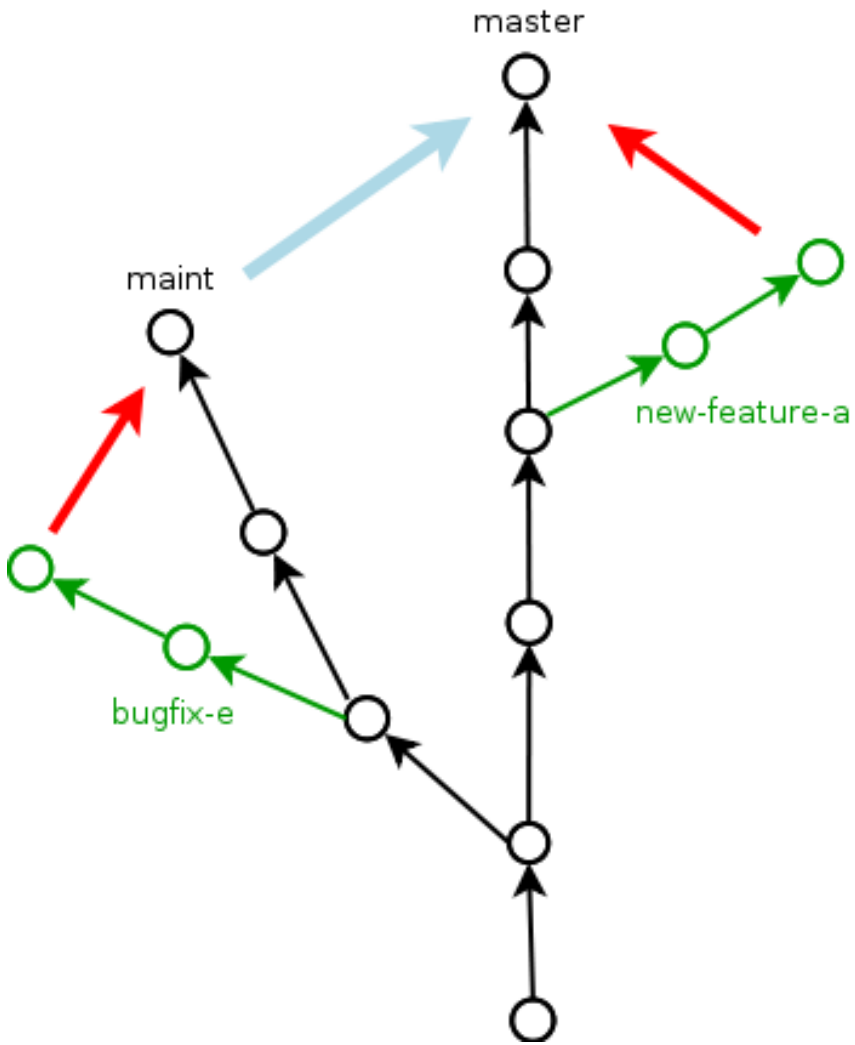
- Some branches use other branches as a parent



Workflow

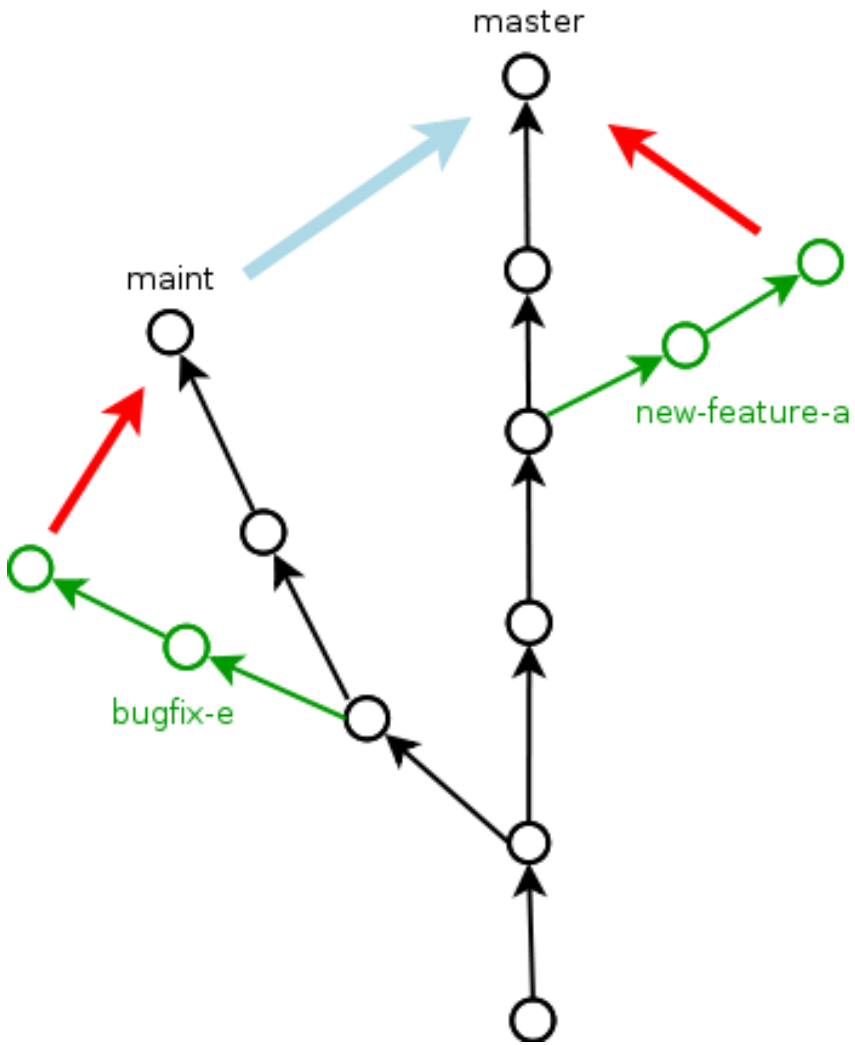
1. Create a **development branch**

2. Make Changes



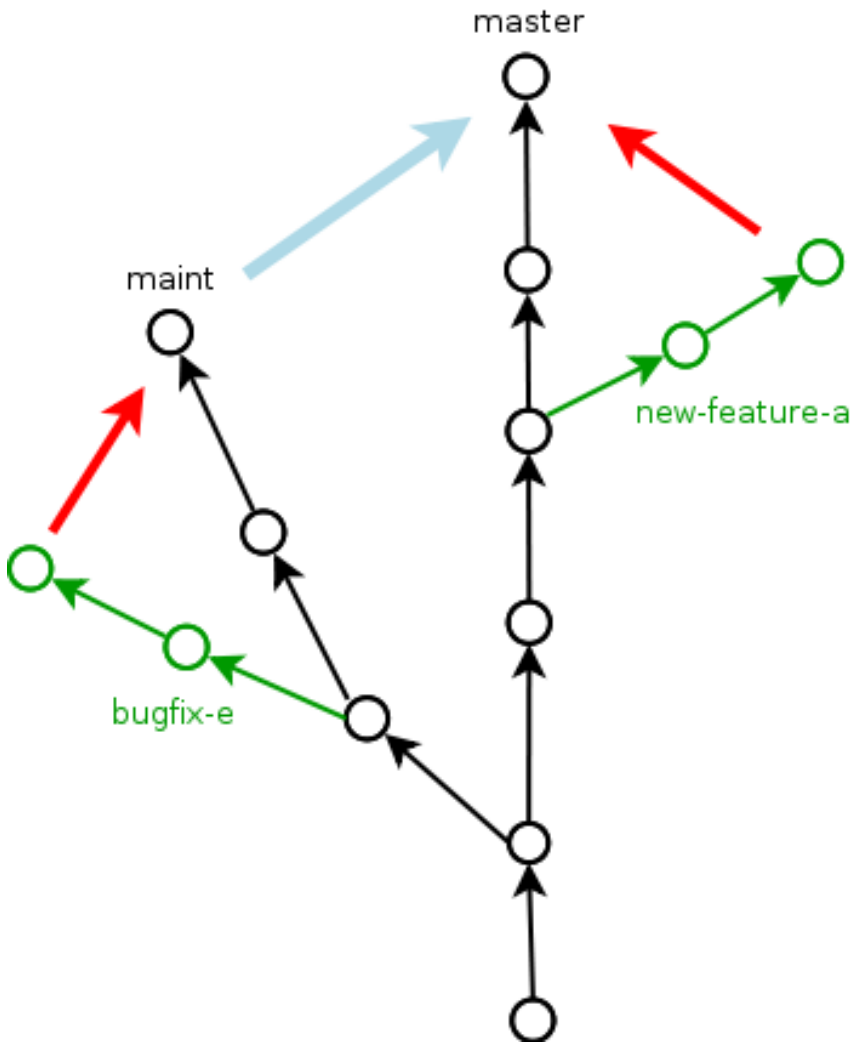
Workflow

1. Create a **development branch**
2. Make Changes
3. Stage File for Commit



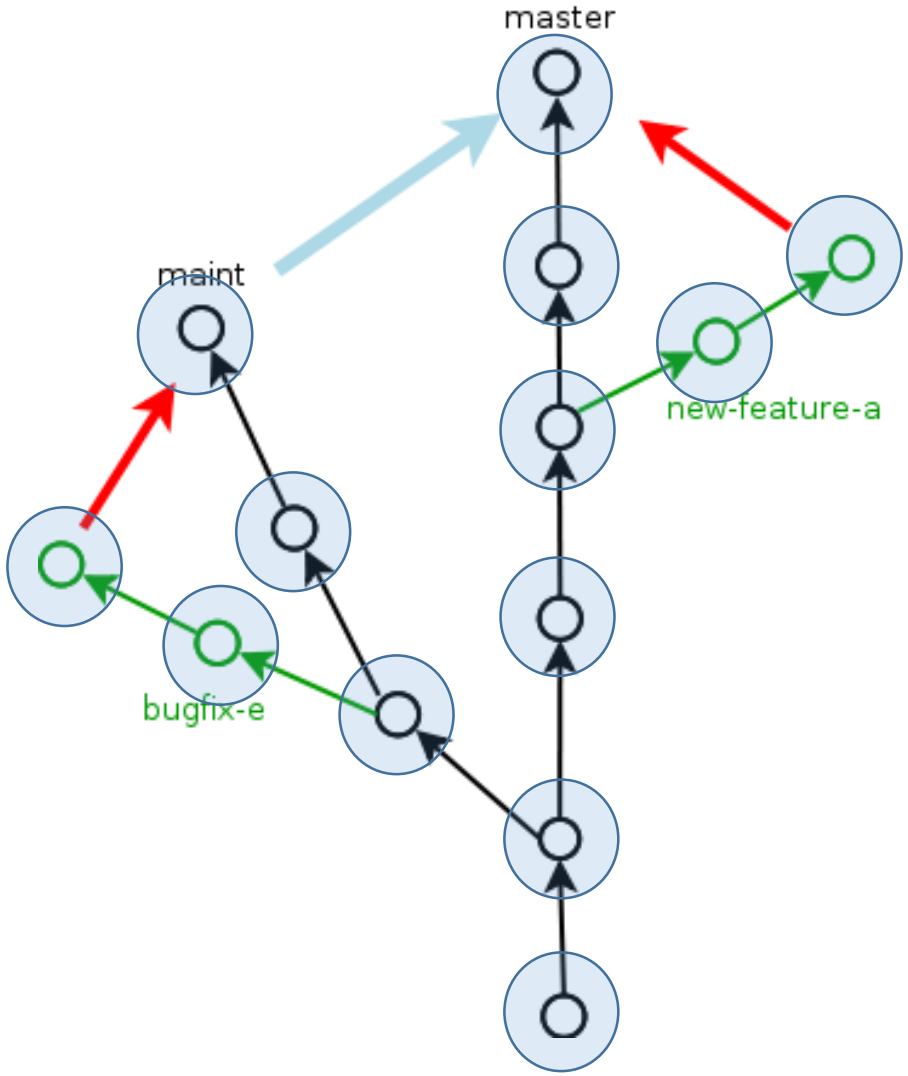
Workflow

1. Create a **development branch**
2. Make Changes
3. Stage File for Commit
4. Commit Change

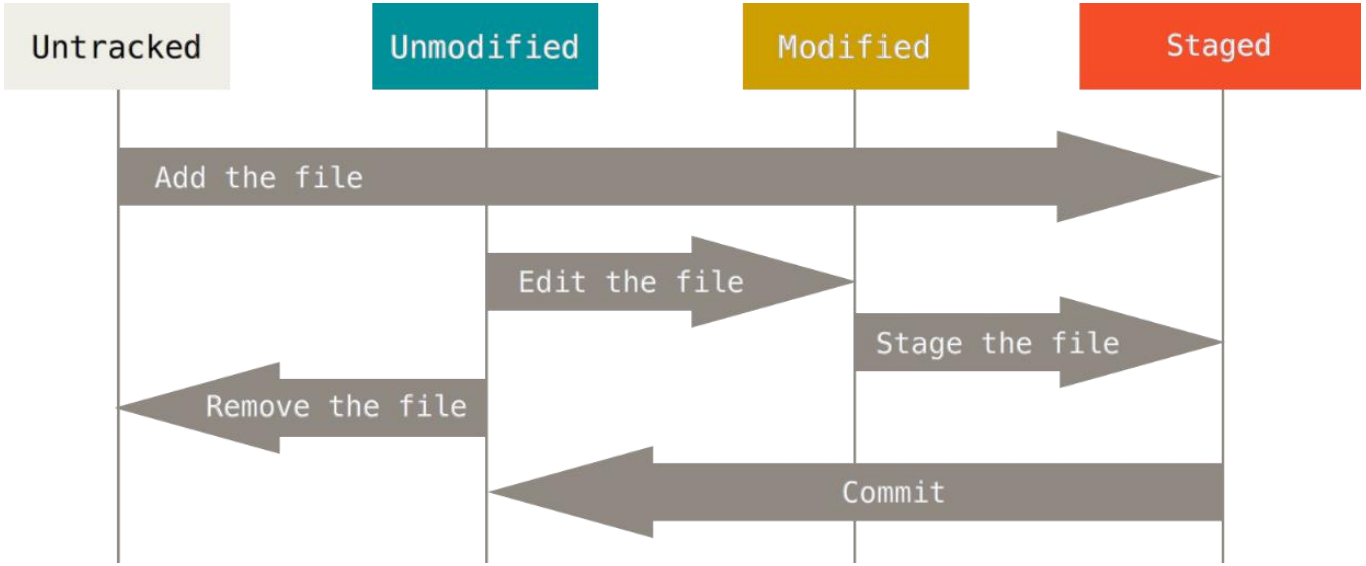


Workflow

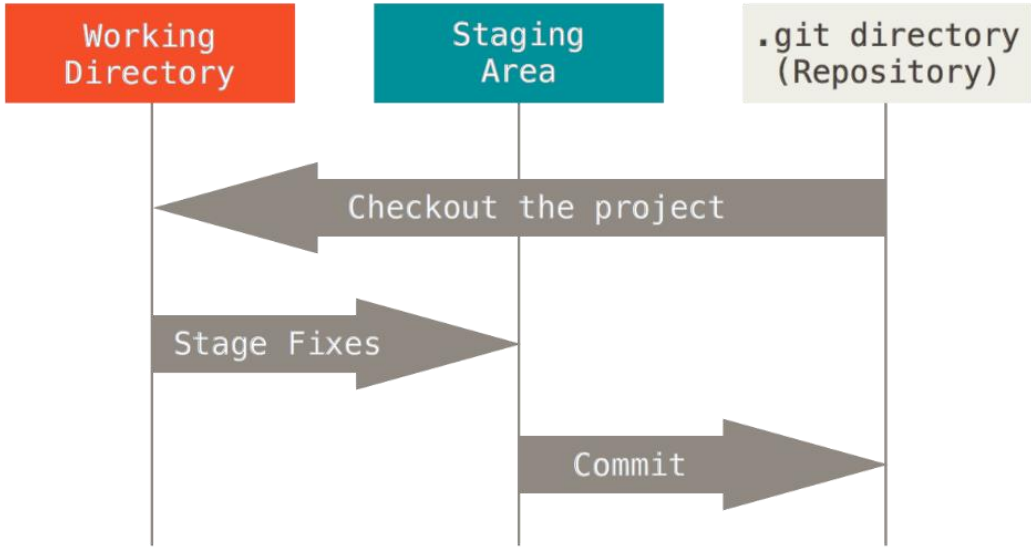
1. Create a **development branch**
2. Make Changes
3. Stage File for Commit
4. Commit Change
   - Each bubble is a commit
   - Each commit is a collection of changed files



Workflow



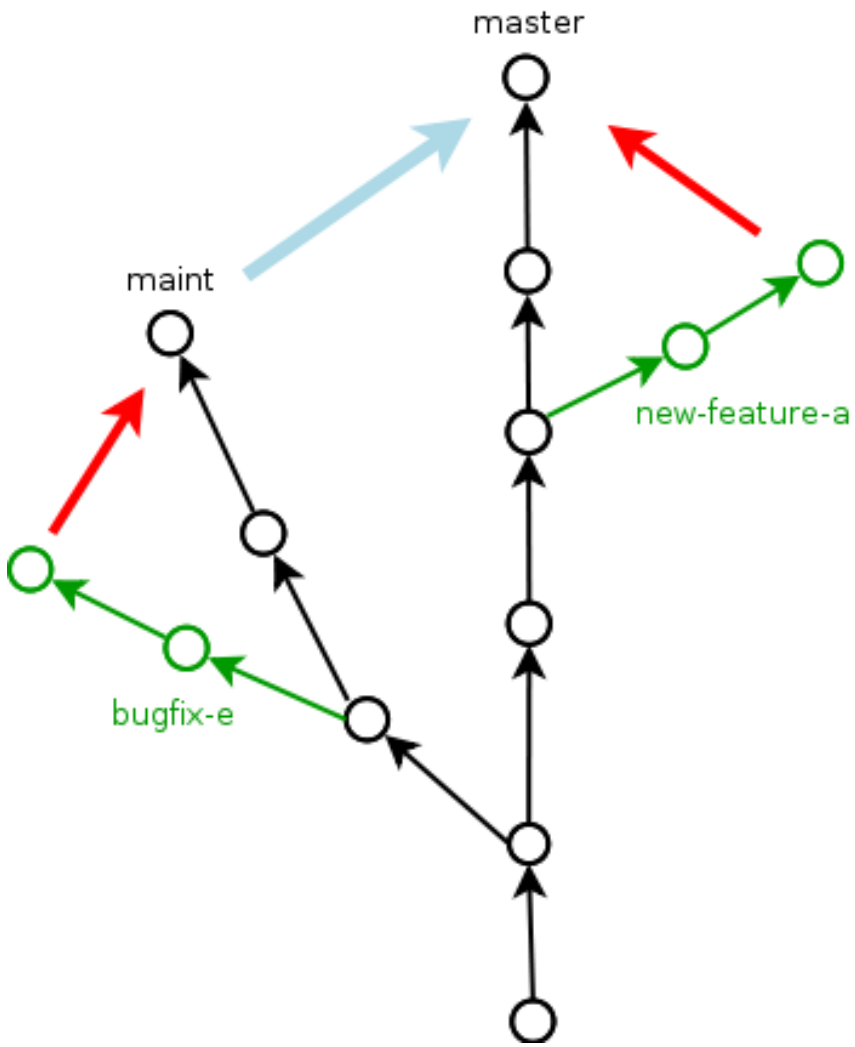
File Classification and the file/git actions



Process of making a commit

Workflow

1. Create a **development branch**
2. Make Changes
3. Stage File for Commit
4. Commit Change
5. Push Change to Remote



Conflicts

- Conflicts can occur when collaborating on a project
- **Conflict Resolution**: Process of resolving conflicting changes between to change-sets
- Tools related to merging and conflict resolution:
  - git merge
  - git rebase
  - git diff

Importance of Version Control

- Historical Analysis of code changes
- Metadata helps track progress (Task Management/Bug Tracking)
- Advocates Collaboration and Co-Development of code