git and GitHub

Version Control System (VCS)

git is a VCS

- Code repository management for individuals and teams
- Multiple versions, conflict resolution, rollback, etc.

git enables teams to work on projects **safely** and **efficiently**.

git

Developed by Linus Torvalds for Linus Torvalds

- Powerful and robust, but has some quirks

Mastery comes with experience.

git != GitHub

git is a VCS
- A set of tools for managing repositories.
- By default, a repository is local.

GitHub is a hosting service
- Implements git
- Provides additional tools (Pages, Issues, etc.)
- Other git hosting services exist (e.g. BitBucket)

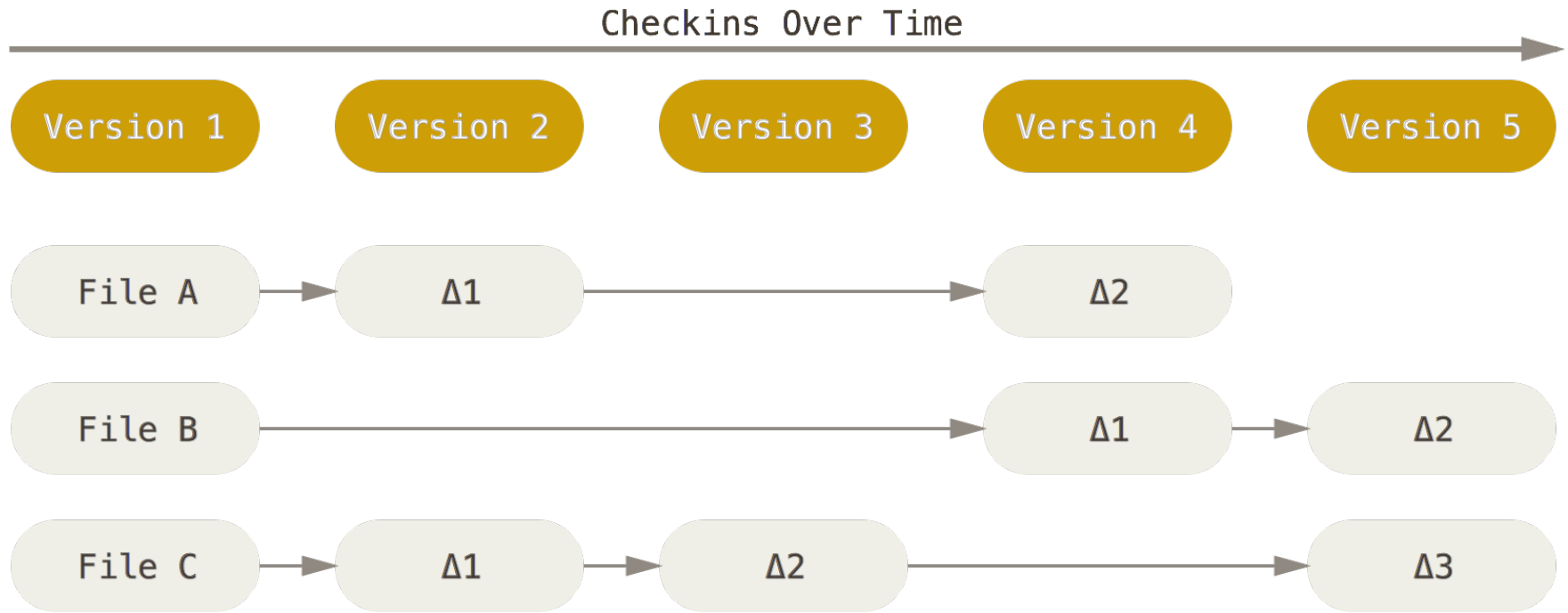
Let’s talk about git

To use git effectively, it’s important to know how it works.

- git is fundamentally different from most other VCS

git vs. Conventional VCS

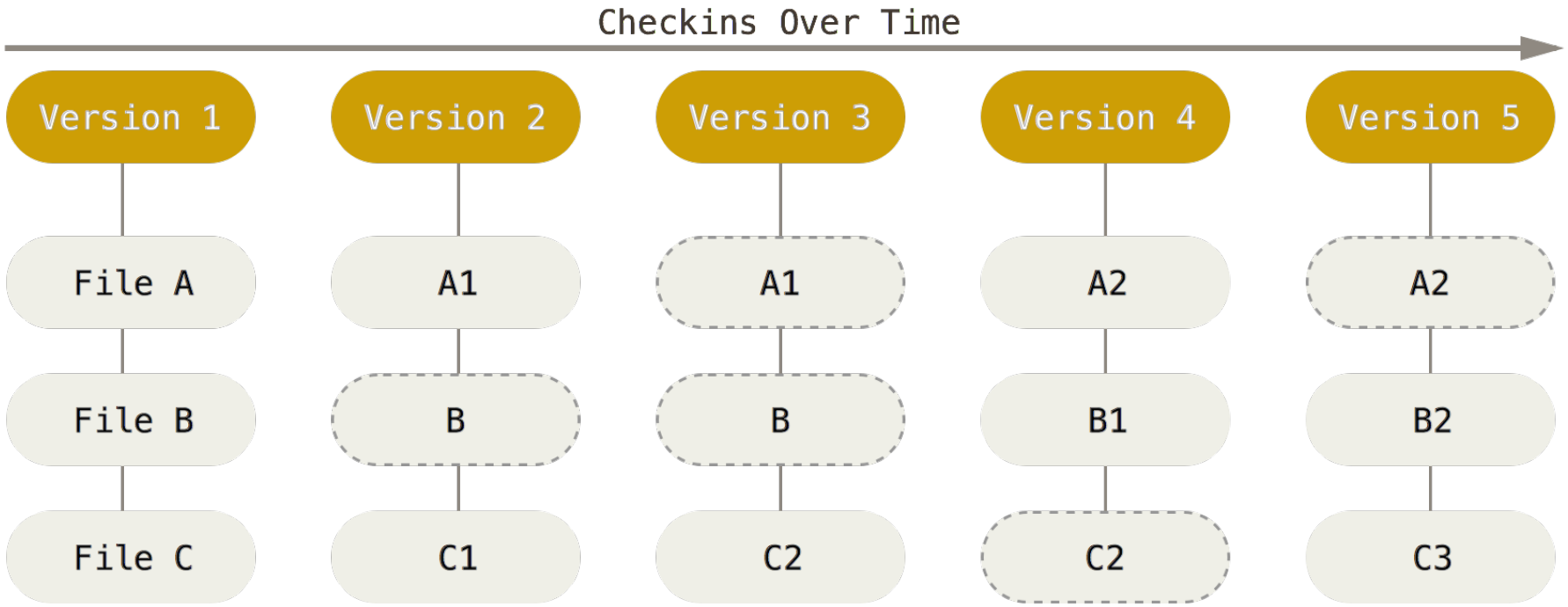
Most other VCS manage a repository using a list of changes.



https://git-scm.com/book/en/v2/Getting-Started-Git-Basics

git vs. Conventional VCS

git stores data as a series of snapshots and links.



https://git-scm.com/book/en/v2/Getting-Started-Git-Basics

git is Local

- Your local copy contains the entire repository history
- Creating branches, commits, merges, etc. are all local operations
- You can safely work offline and push later

Three States of git

- Committed
  - Data is safely stored in database
- Modified
  - File is changed and not yet committed
- Staged
  - Modified file is marked for commit

Basic git workflow

The basic Git workflow goes something like this:

- You modify files in your working tree.
- You stage the files, adding snapshots of them to your staging area.
- You do a commit, which takes the files as they are in the staging area and stores that snapshot permanently to your Git directory.

https://git-scm.com/book/en/v2/Getting-Started-Git-Basics

git Operations

These basic operations can be used to create a new repository

- init
  - Initialize a local git repository (.git file)
- add
  - Tell git to include a file as part of the repository
- commit
  - Saves staged snapshots to repository on current branch

git Operations

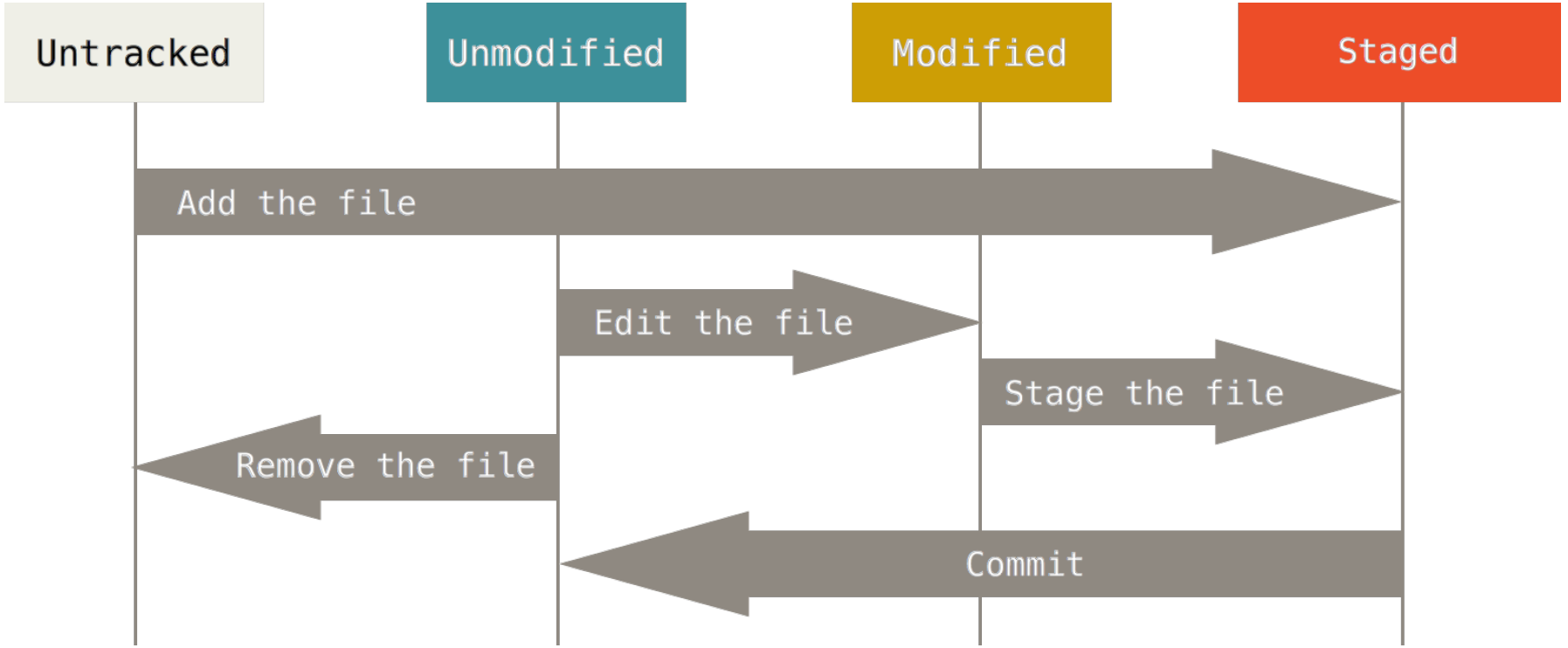
Cloning a repository

- clone
  - Gets a copy of an existing repository from URL

https://git-scm.com/book/en/v2/Git-Basics-Getting-a-Git-Repository

Making Changes

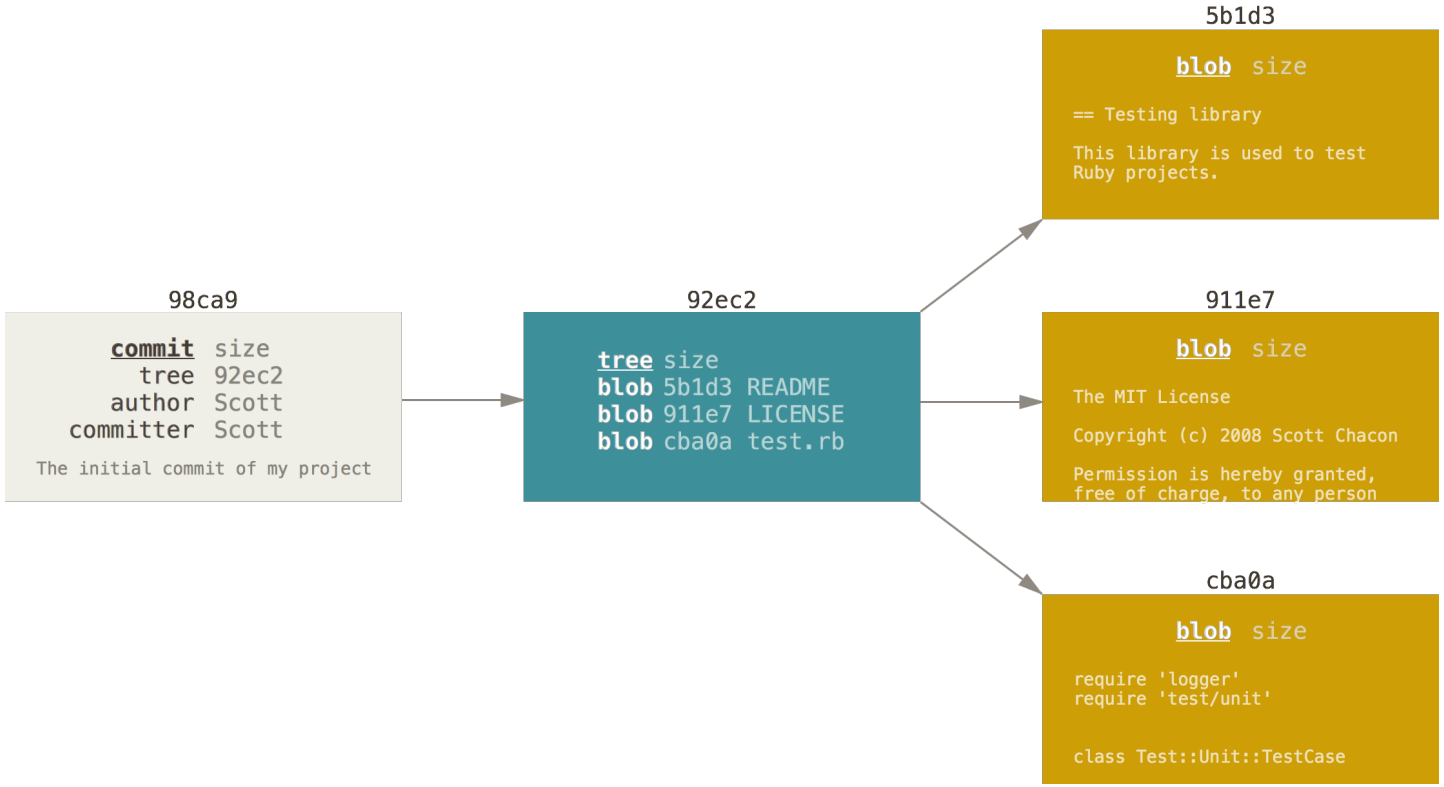
Typical workflow while working within a branch



https://git-scm.com/book/en/v2/Git-Basics-Recording-Changes-to-the-Repository

Branching

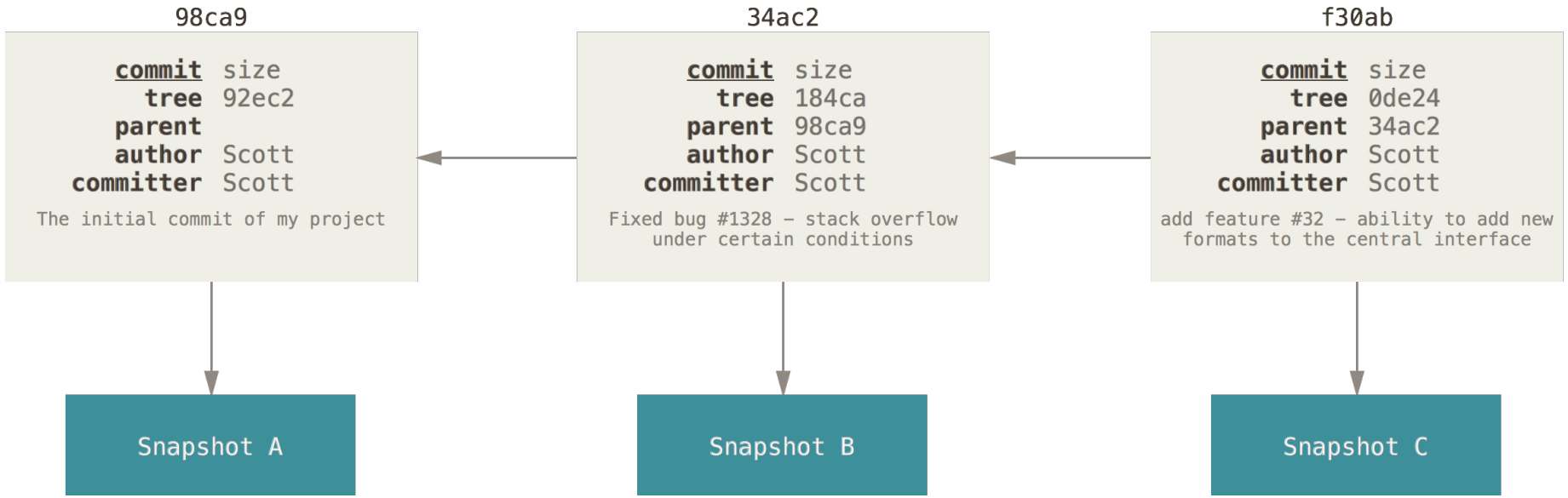
Commits point to a snapshot tree.



https://git-scm.com/book/en/v2/Git-Branching-Branches-in-a-Nutshell

Branching

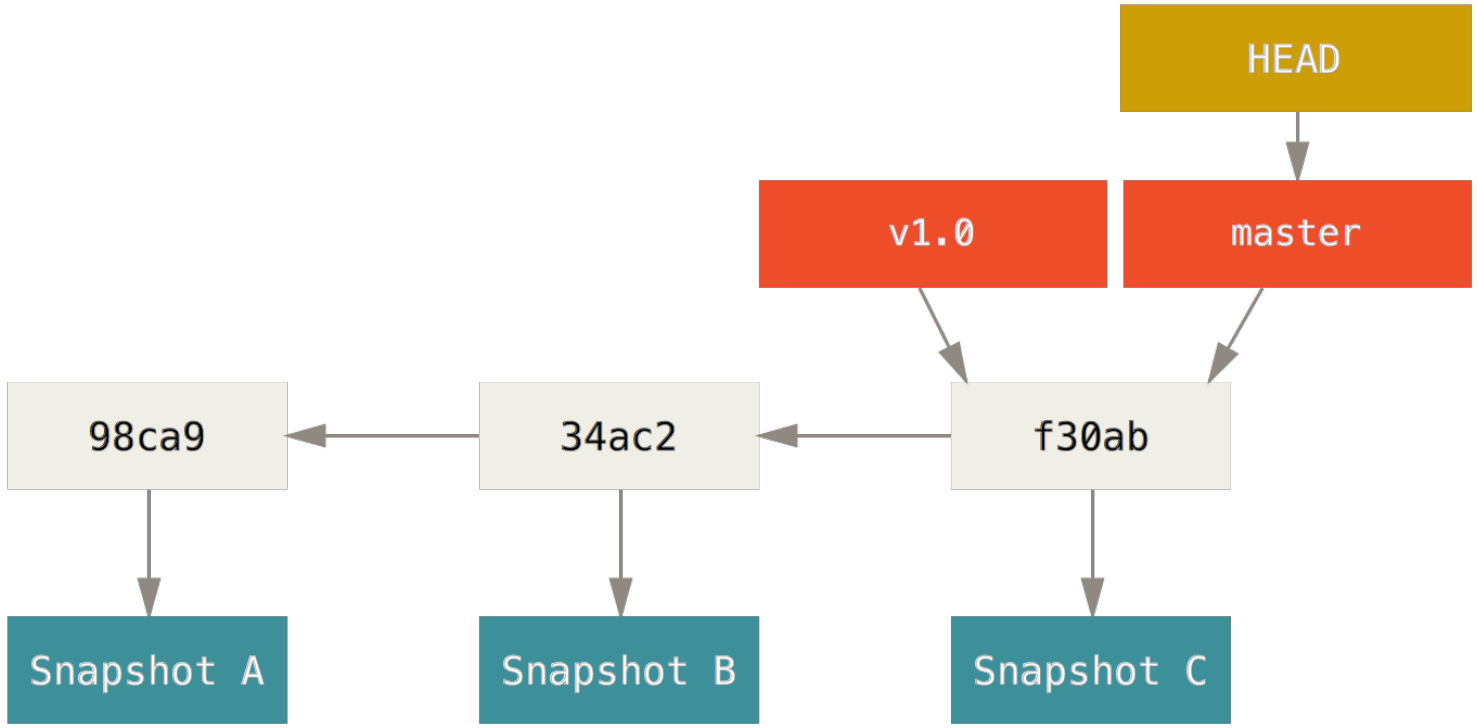
When you make changes, a new snapshot is created along with a new commit pointer.



https://git-scm.com/book/en/v2/Git-Branching-Branches-in-a-Nutshell

Branching

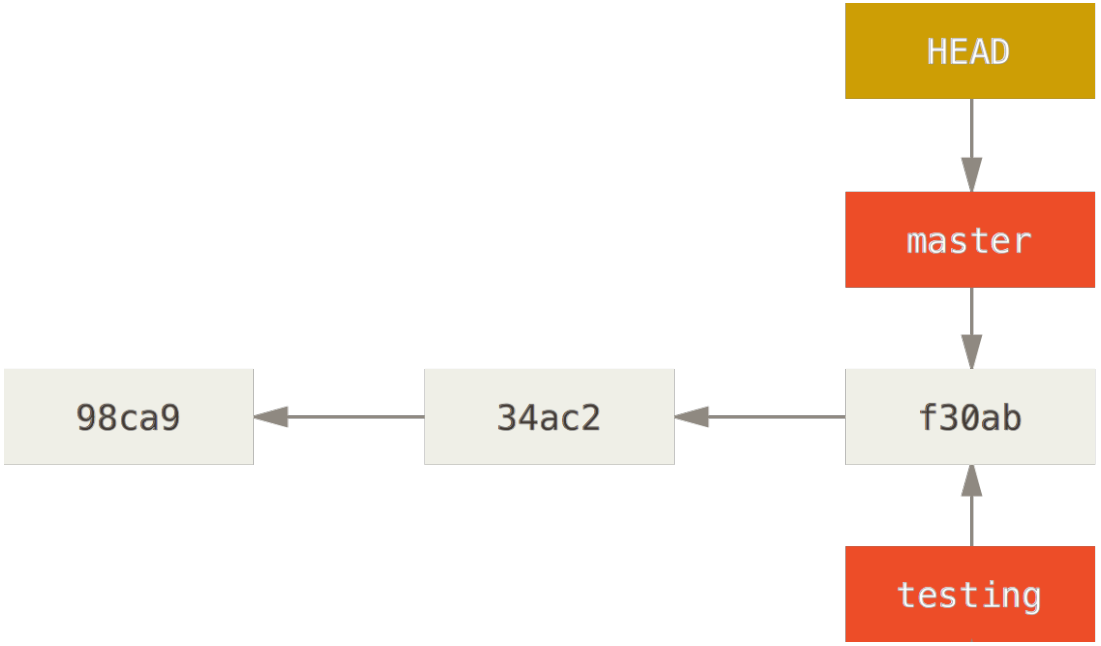
A branch and its commit history.



https://git-scm.com/book/en/v2/Git-Branching-Branches-in-a-Nutshell

Branching

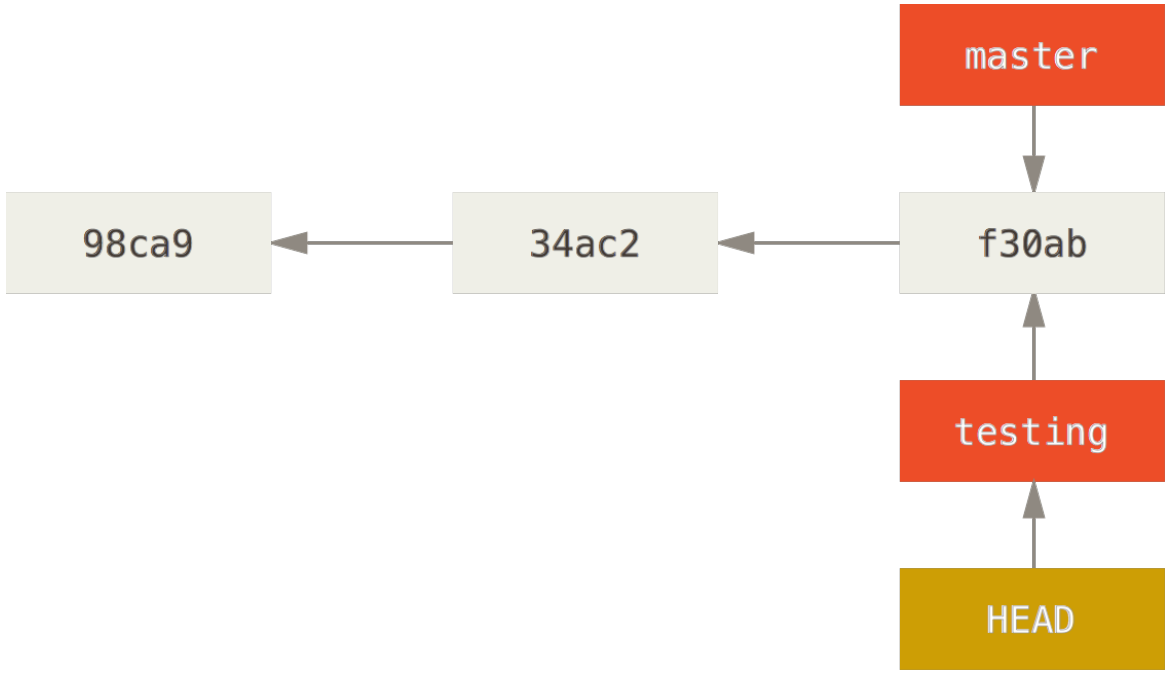
Assuming the current branch (HEAD) is pointing to master, suppose we have created a new branch called testing.



https://git-scm.com/book/en/v2/Git-Branching-Branches-in-a-Nutshell

Branching

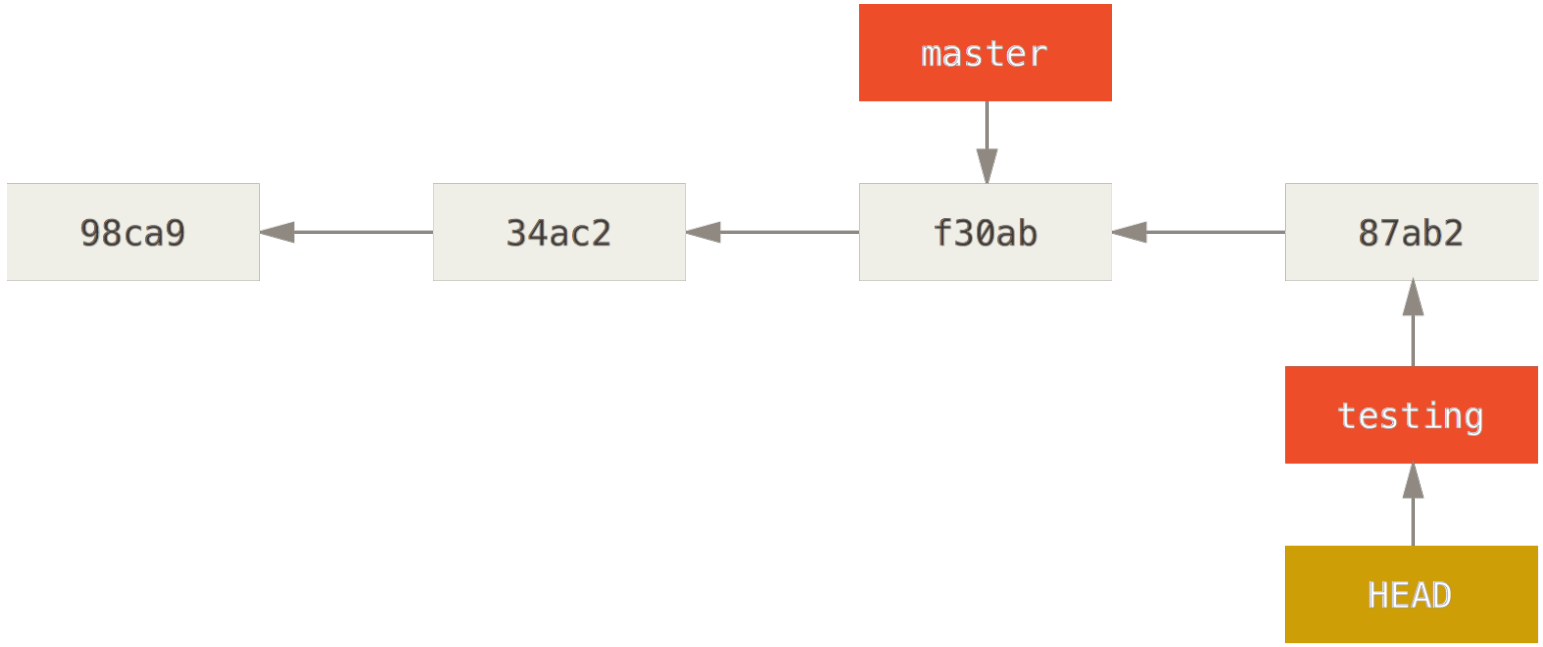
To change branches, we use the checkout command.



https://git-scm.com/book/en/v2/Git-Branching-Branches-in-a-Nutshell

Branching

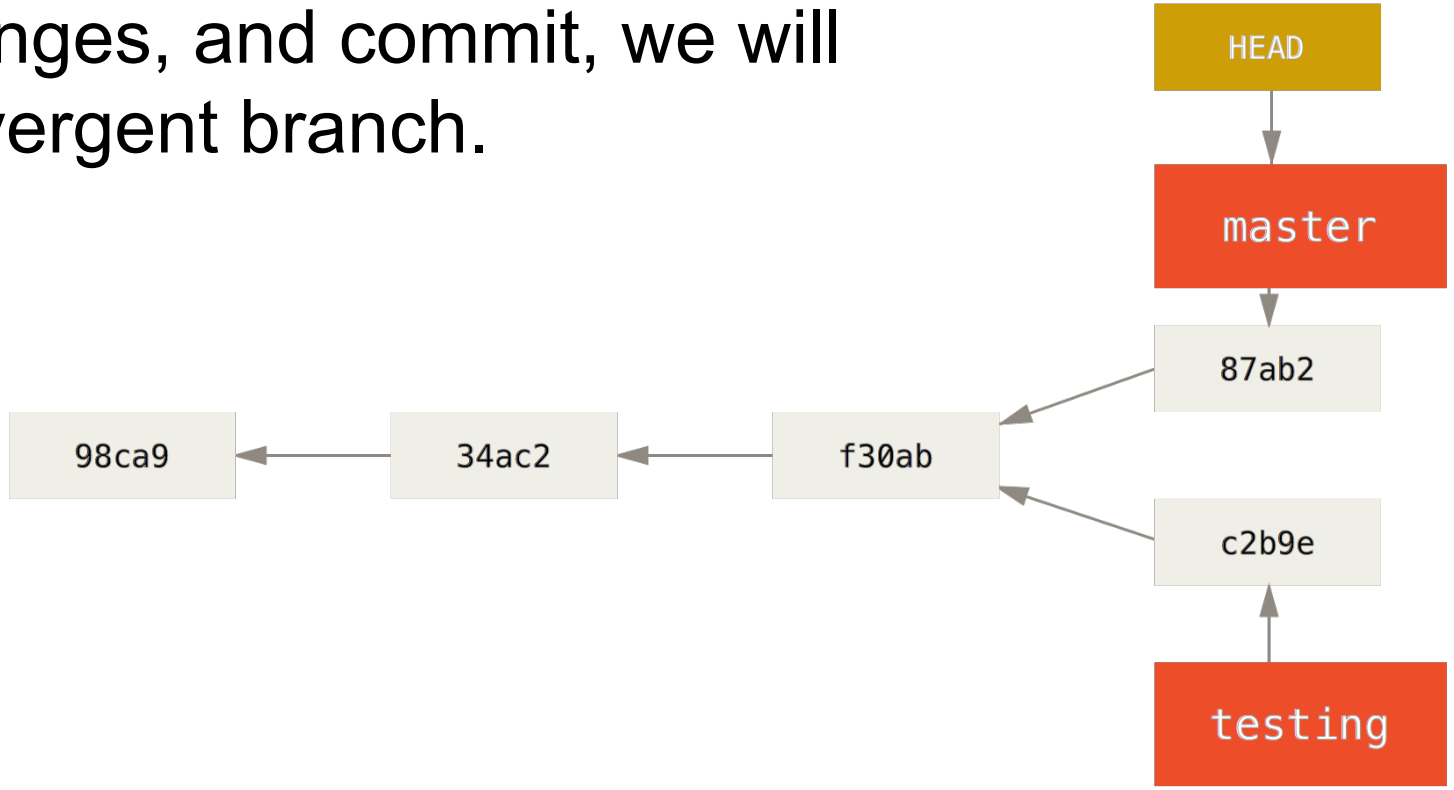
Now if we make changes to the testing branch and commit, a new snapshot is created.



https://git-scm.com/book/en/v2/Git-Branching-Branches-in-a-Nutshell

Branching

Then if we checkout back to master, make changes, and commit, we will have a divergent branch.



https://git-scm.com/book/en/v2/Git-Branching-Branches-in-a-Nutshell

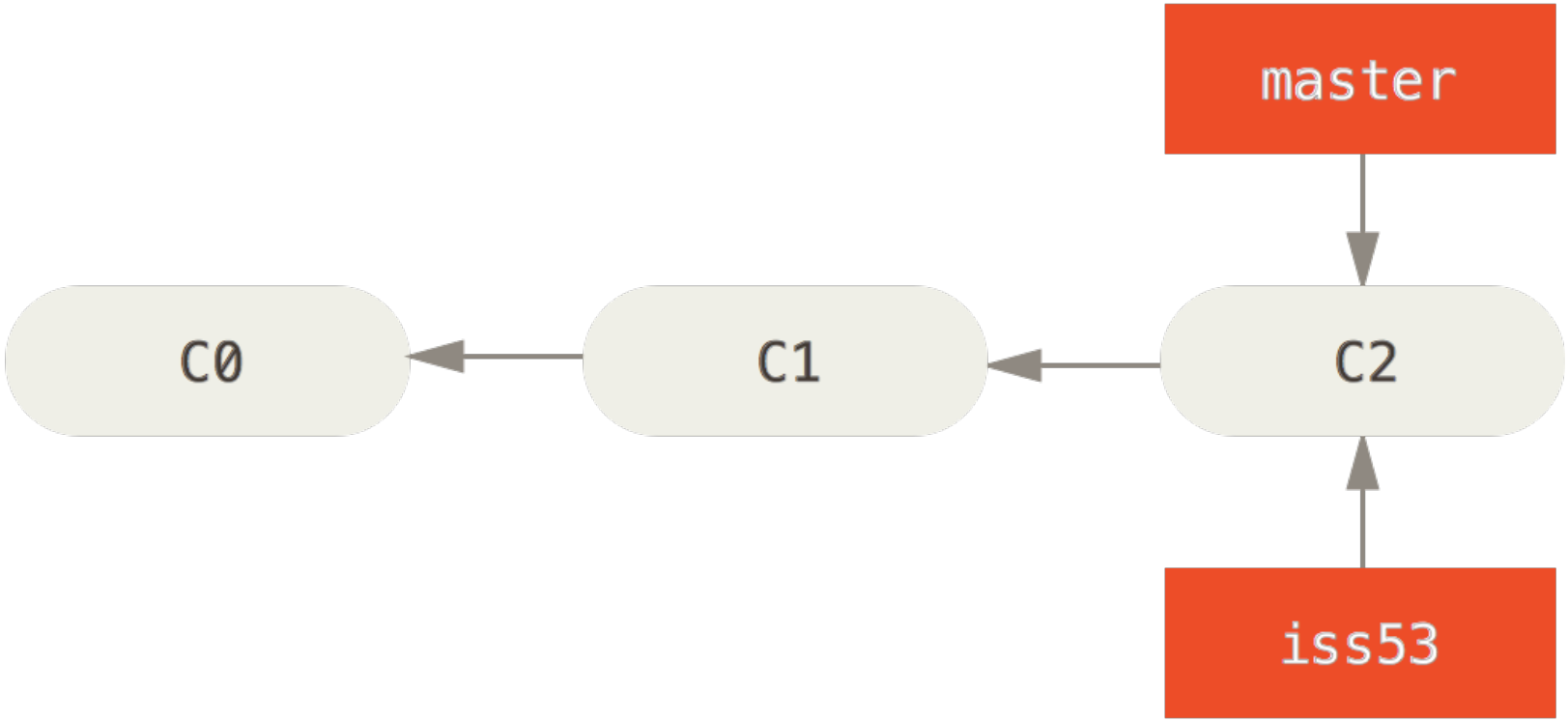
Merging

Merging is the process of combining two branches into a new snapshot and should be done with care.

Good branching strategy can make merging easier.

Merging

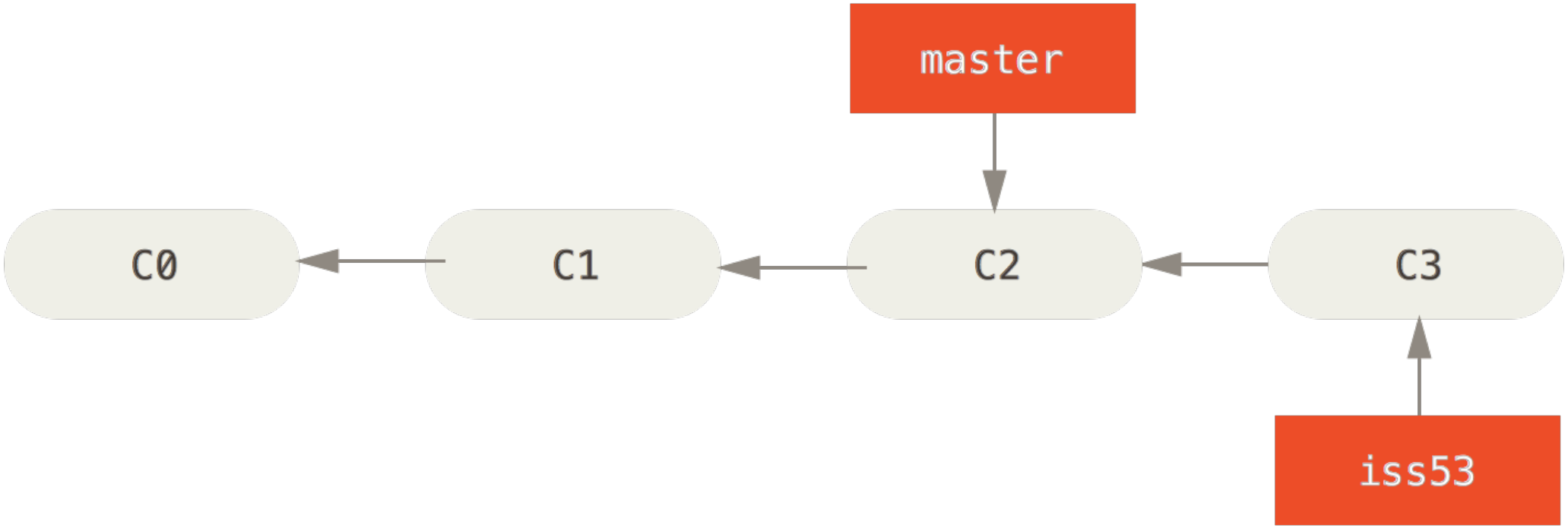
Suppose we have been working on a master branch and decided to create a new branch for resolving an issue.



https://git-scm.com/book/en/v2/Git-Branching-Basic-Branching-and-Merging

Merging

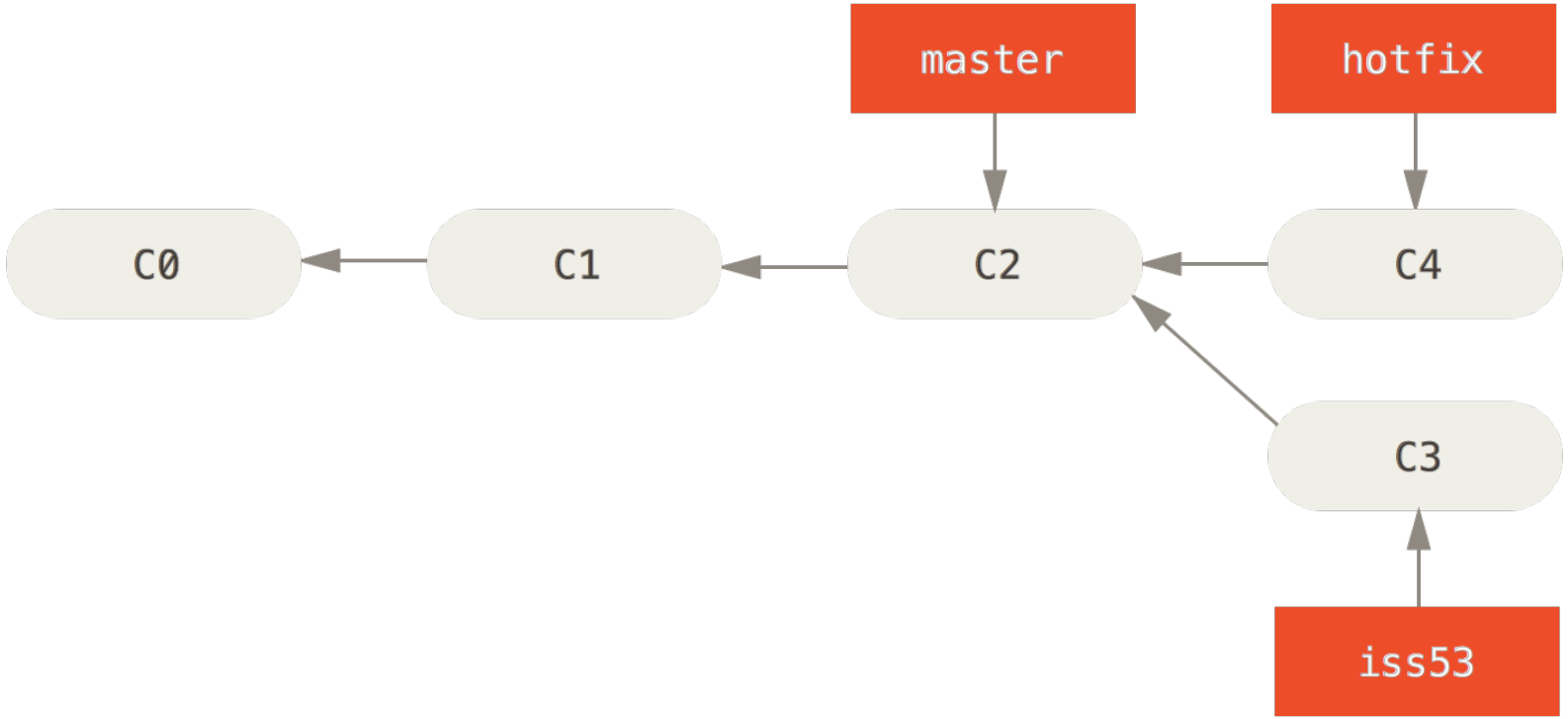
After checking out to iss53, we make some changes and commit.



https://git-scm.com/book/en/v2/Git-Branching-Basic-Branching-and-Merging

Merging

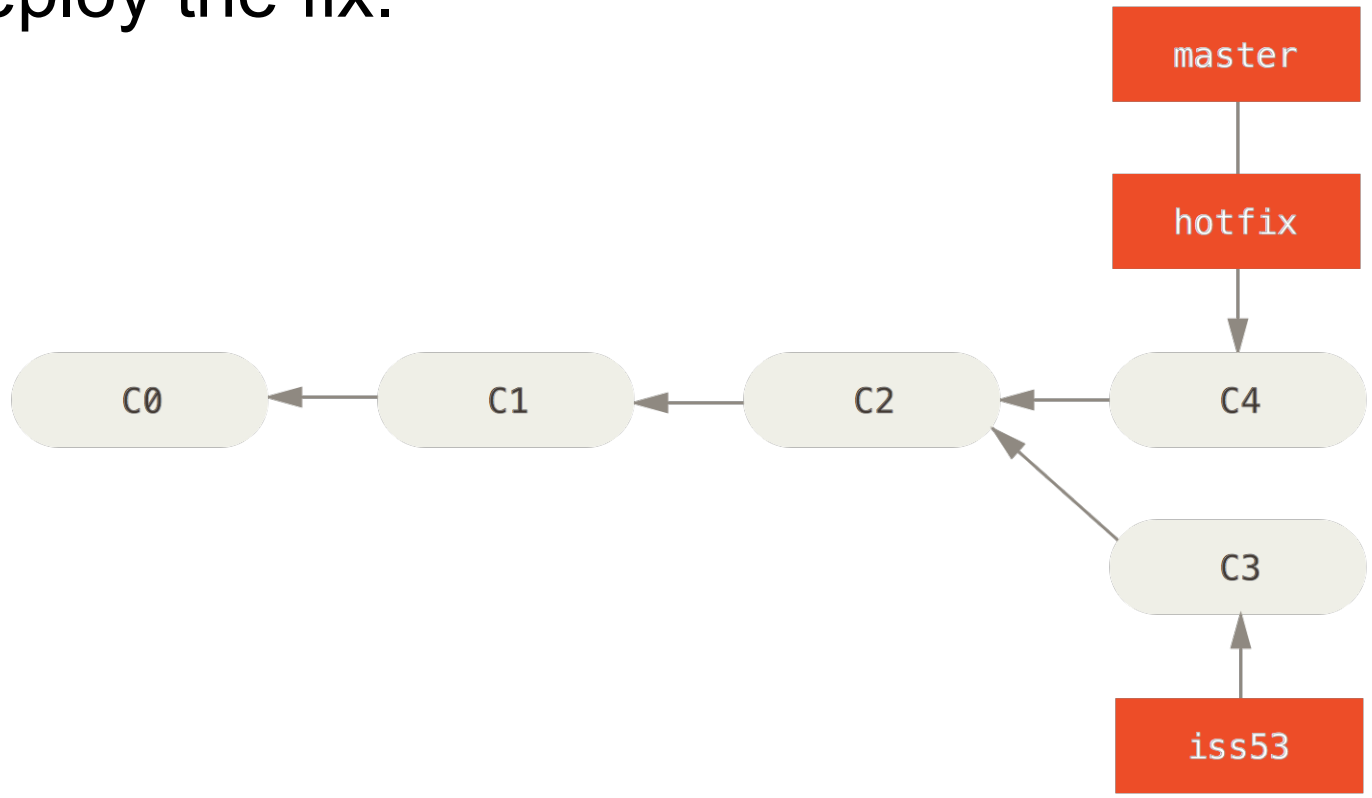
Now suppose there is another, unrelated problem to iss53 that needs immediate attention, and we create a hotfix branch.



https://git-scm.com/book/en/v2/Git-Branching-Basic-Branching-and-Merging

Merging

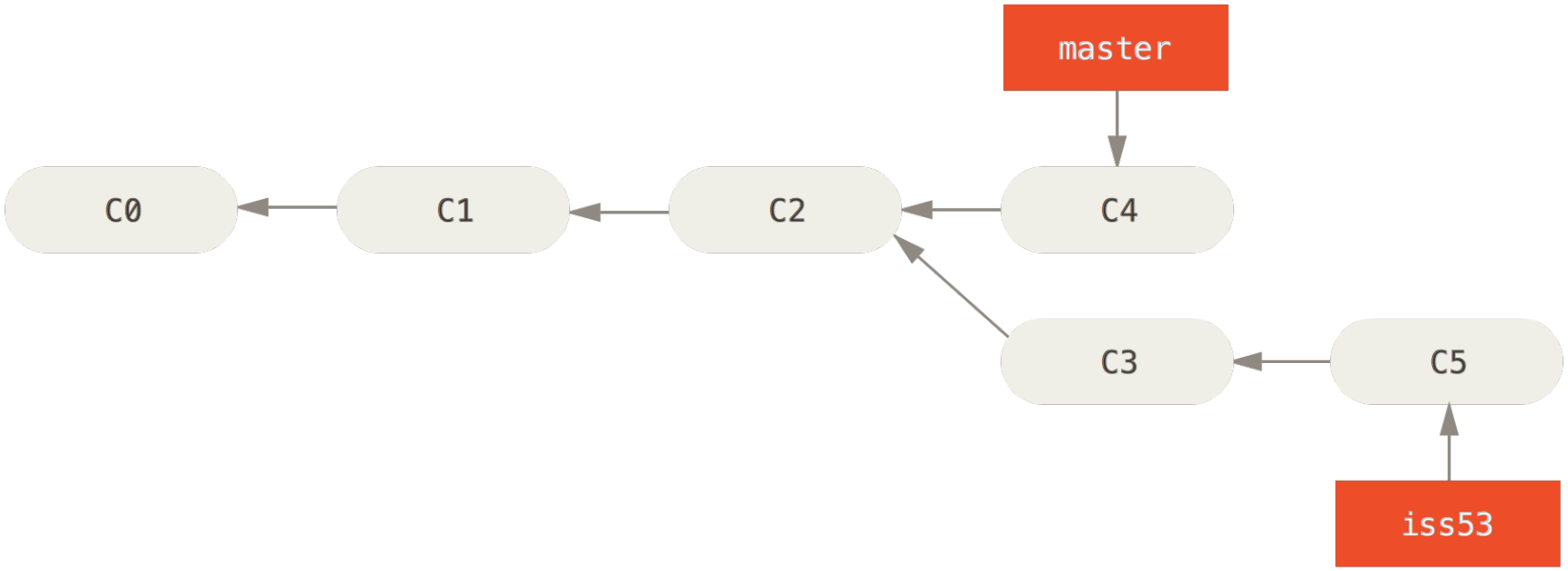
If we are satisfied with the state of hotfix, we can merge it with master to deploy the fix.



https://git-scm.com/book/en/v2/Git-Branching-Basic-Branching-and-Merging

Merging

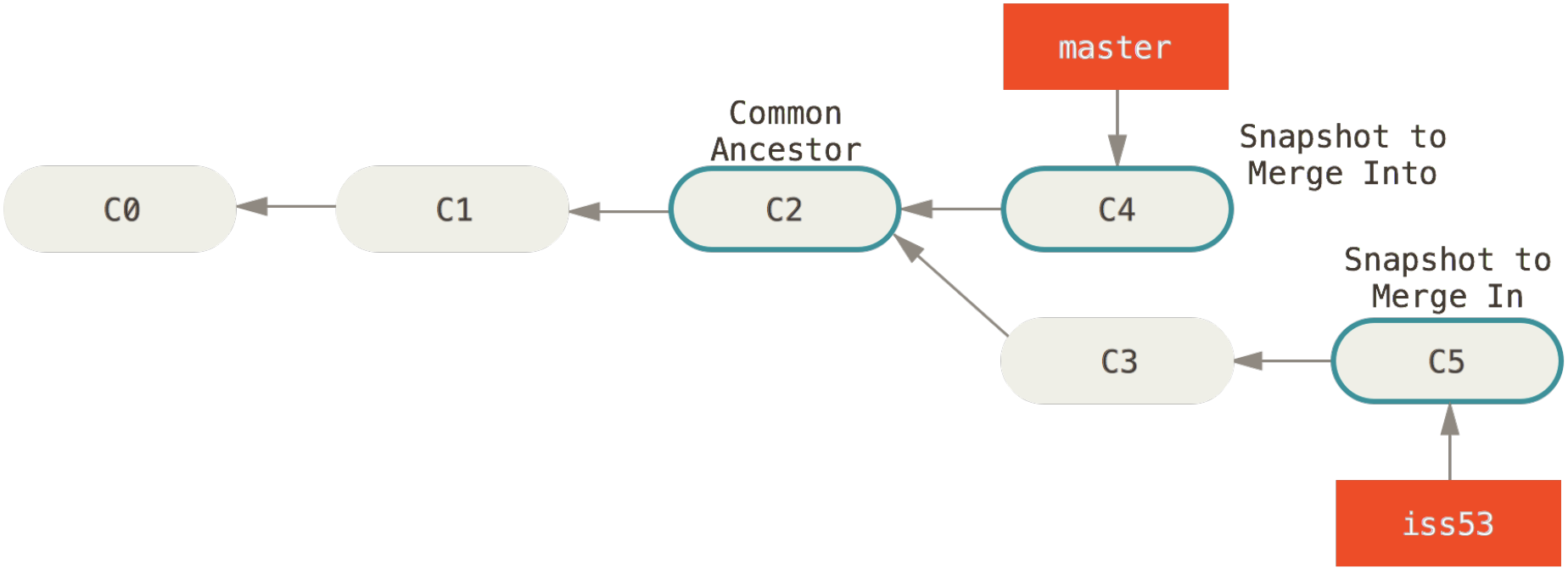
Now suppose that we continue to work on iss53. We checkout to iss53, make changes, and commit.



https://git-scm.com/book/en/v2/Git-Branching-Basic-Branching-and-Merging

Merging

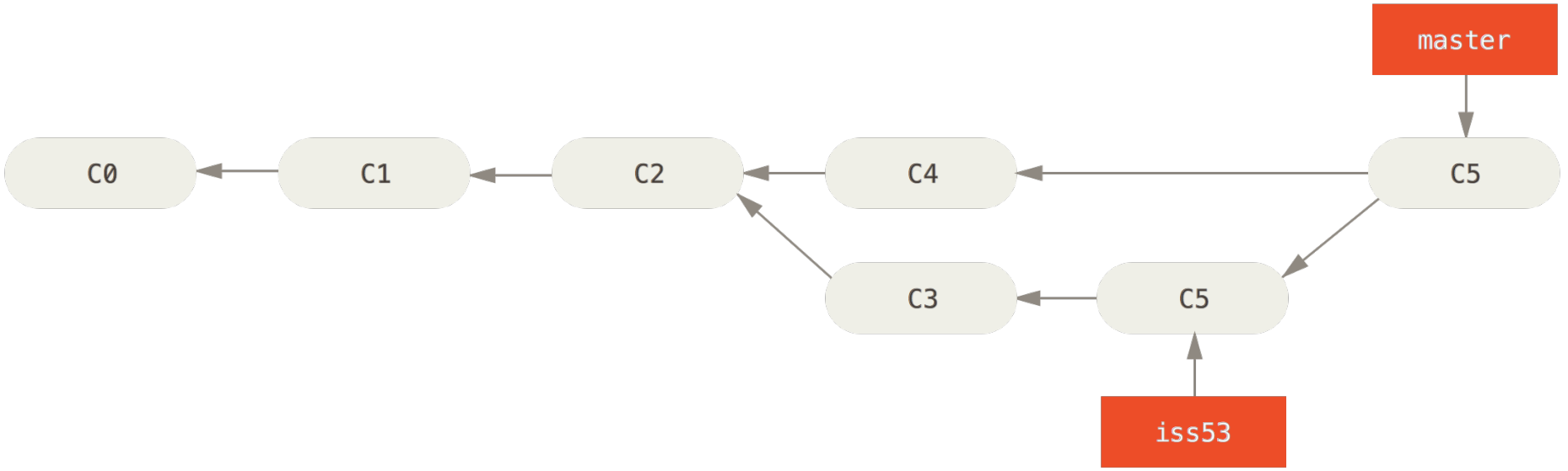
To merge master and iss53, git needs to handle the fact that the two branches may have conflicting changes.



https://git-scm.com/book/en/v2/Git-Branching-Basic-Branching-and-Merging

Merging

After conflict resolution, a new snapshot is created and master is set to point to it. This ’merge commit’ is special in that it has two parents: **git is NOT A TREE**



https://git-scm.com/book/en/v2/Git-Branching-Basic-Branching-and-Merging

Merge Conflicts

Resolving merge conflicts can be difficult. Be careful and attentive when merging. You need to understand your code in order to merge successfully.

Interfaces

In order to use git, there are a variety of available interfaces. To learn how to use git on the command line, there are many tutorials available online, in addition to the **documentation**.

In this course, we will use IntelliJ's git integration extensively.

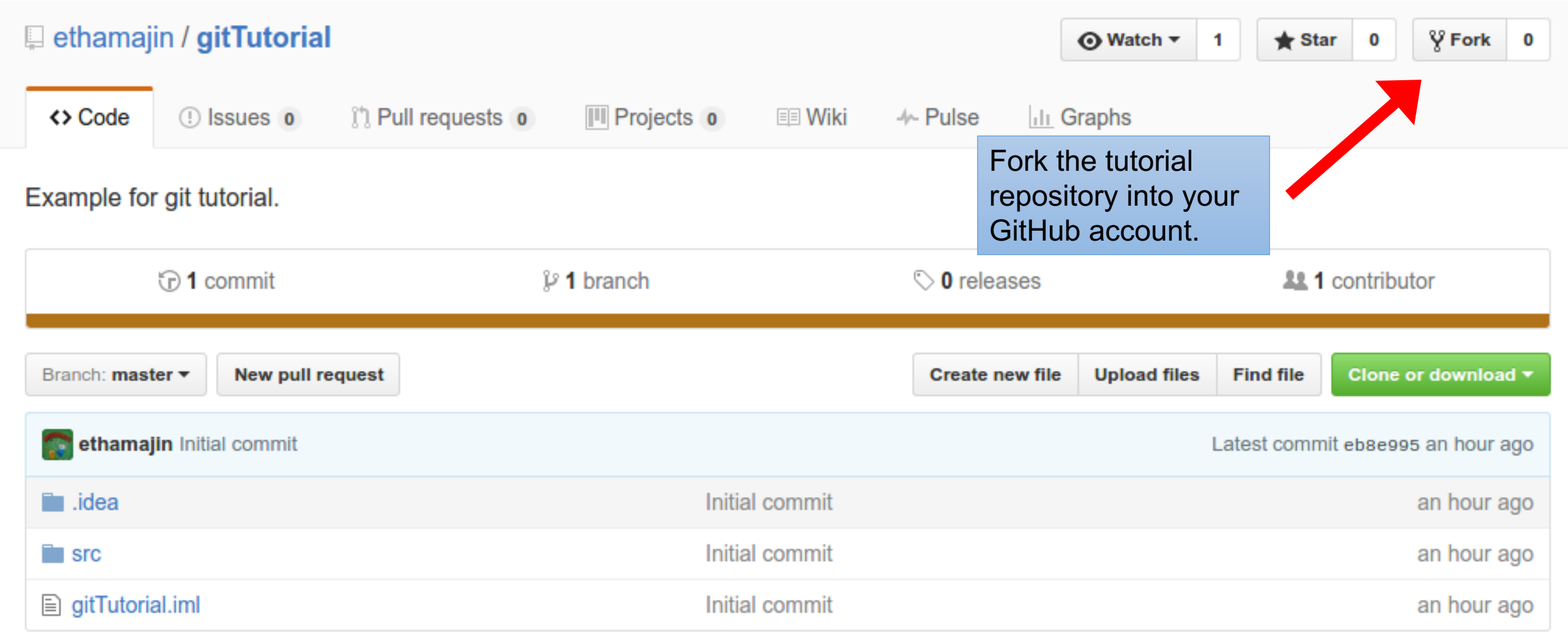
IntelliJ/git By Example

The remainder of this tutorial will focus on git usage in IntelliJ.

We will

- Fork an existing project
- Create a main development branch
- Create two feature branches
- Make changes on each feature branch
- Merge the feature branches with the development branch
- Merge the development branch with master

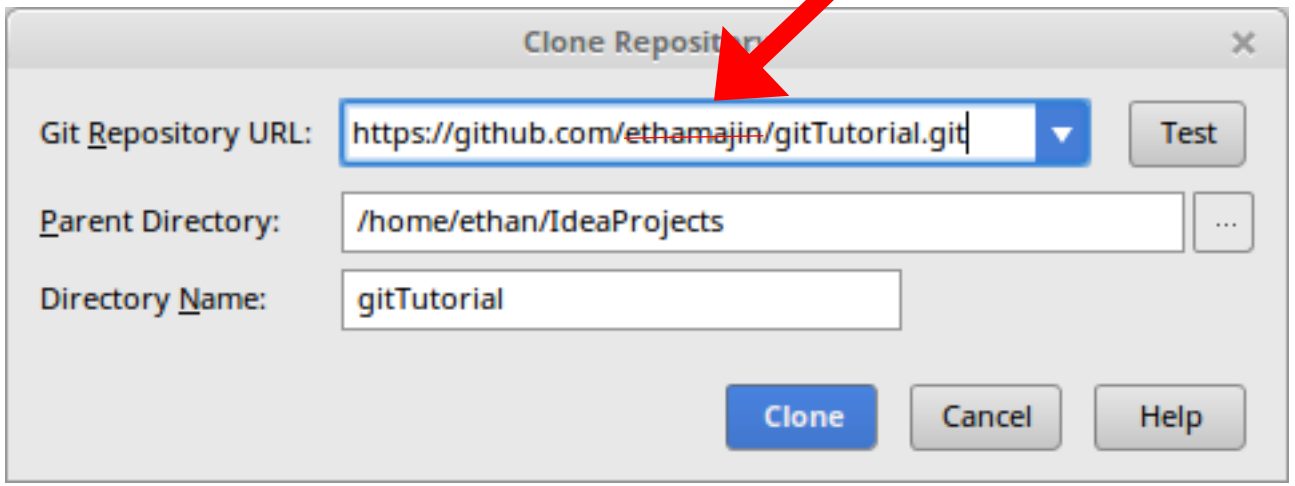
https://github.com/ethamajin/gitTutorial

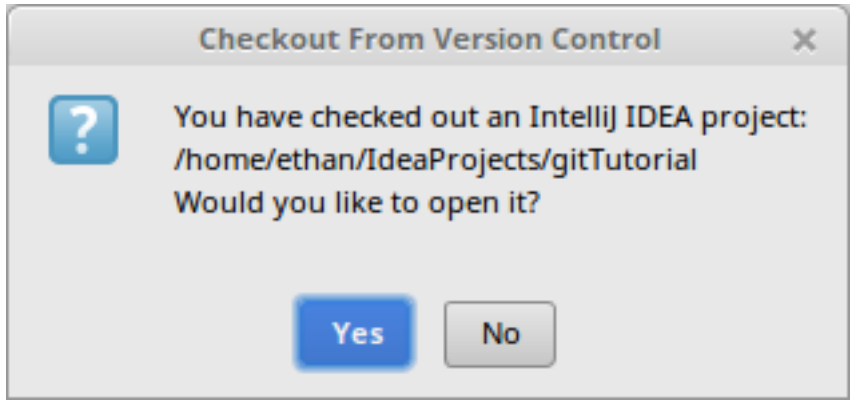


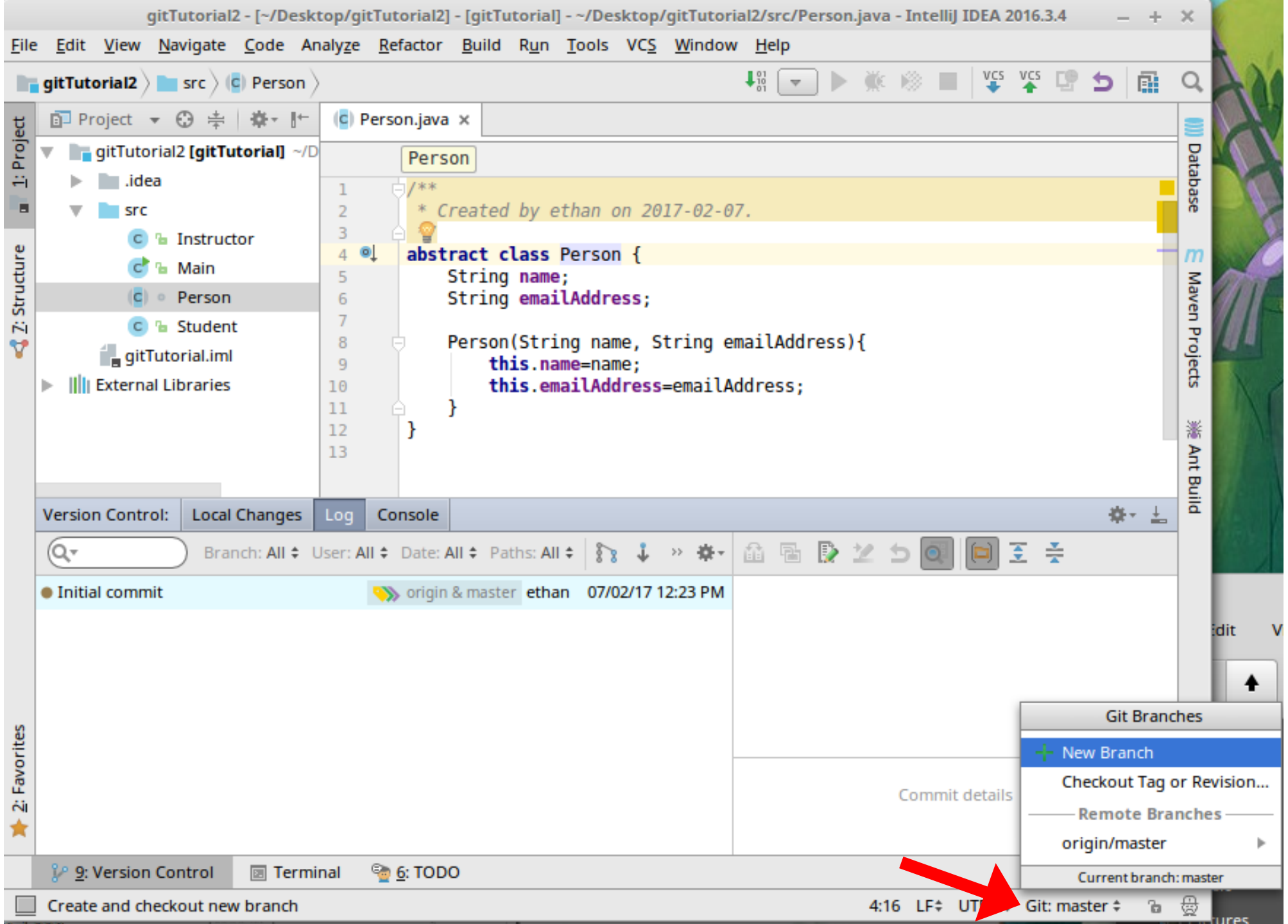

Welcome to IntelliJ IDEA
IntelliJ IDEA
Version 2016.3.4
Create New Project
Import Project
Open
Check out from Version Control
Configure
Get Help

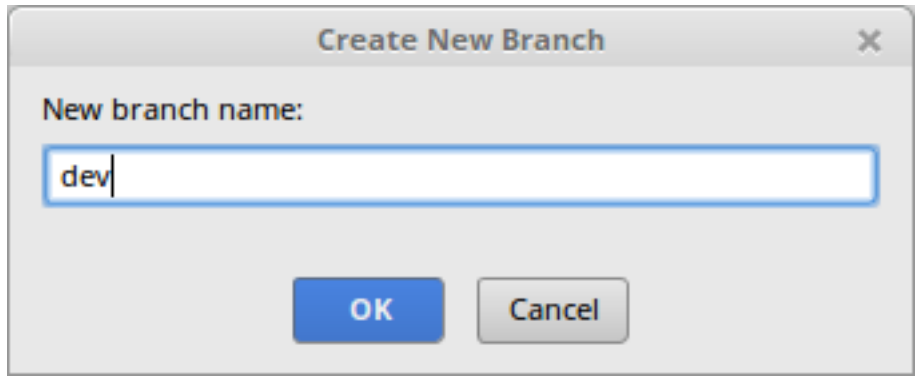

Login to GitHub
Host:
github.com
Auth Type:
Password
Login:
ethamajin
Password:
Do not have an account at github.com? Sign up
Save credentials
Login
Cancel
Help

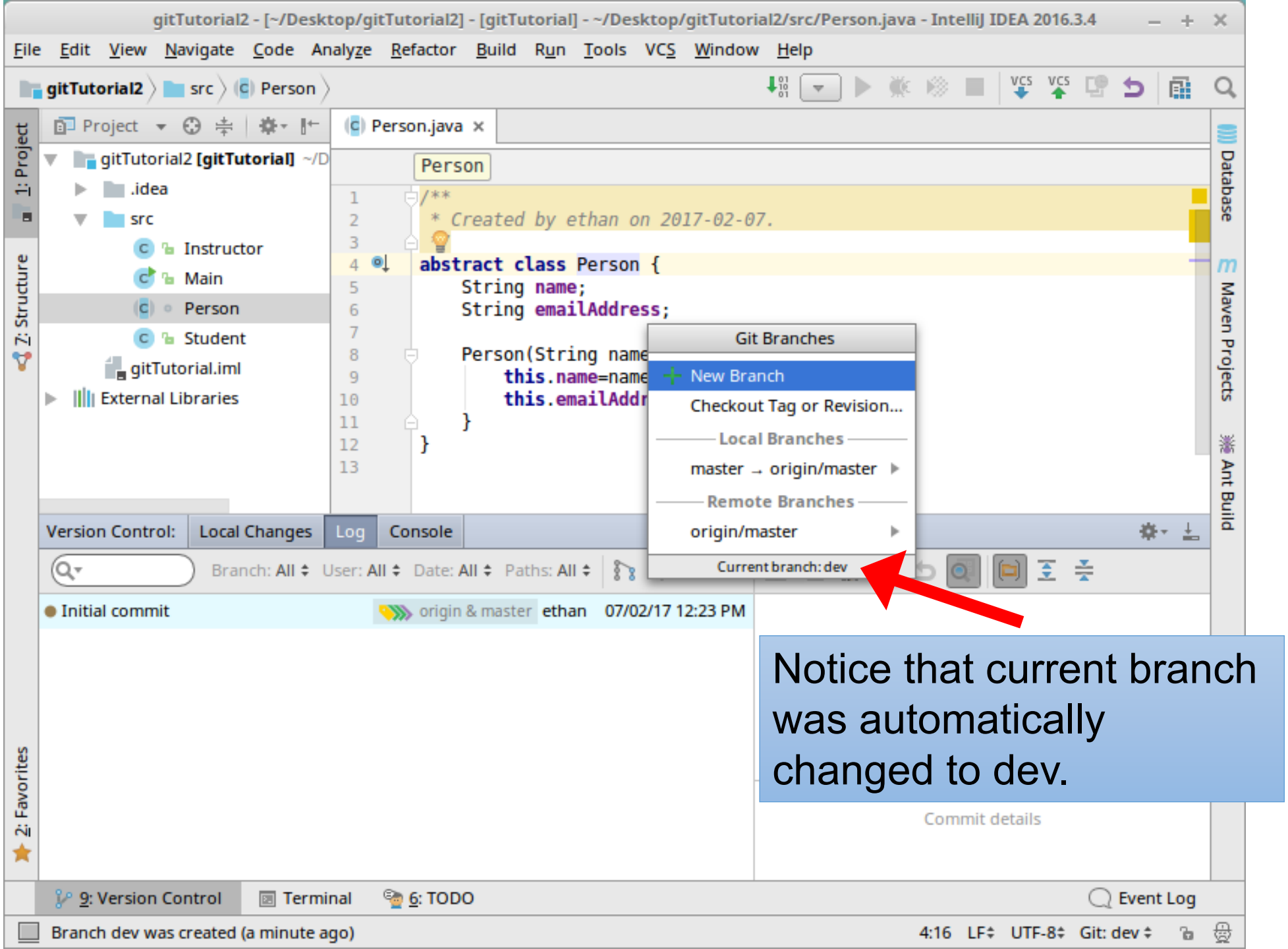
URL for your **forked** repository

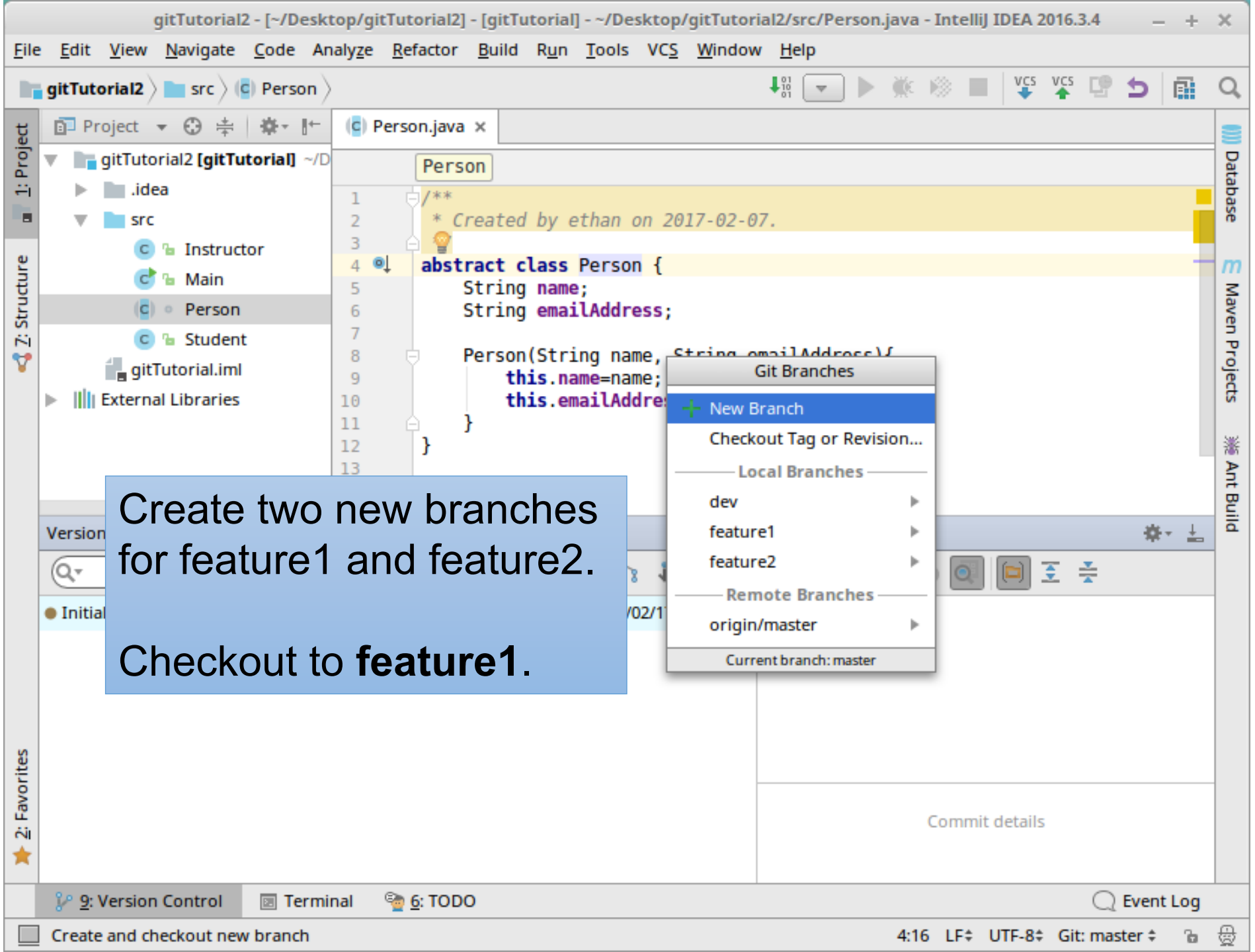


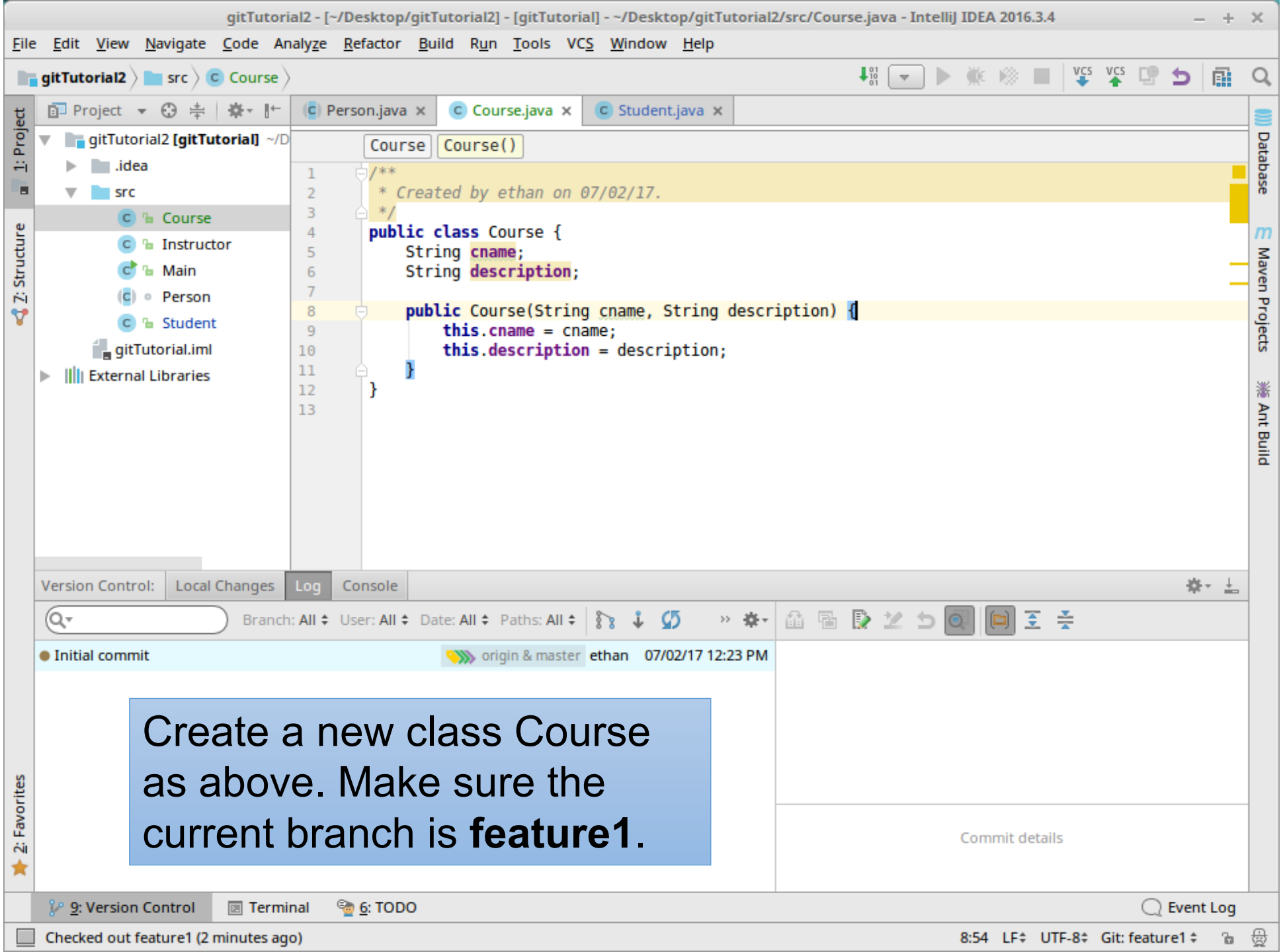

Checkout From Version Control
You have checked out an IntelliJ IDEA project:
/home/ethan/IdeaProjects/gitTutorial
Would you like to open it?
Yes
No


gitTutorial2 - [~/Desktop/gitTutorial2] - [gitTutorial] - ~/Desktop/gitTutorial2/src/Person.java - IntelliJ IDEA 2016.3.4
File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help
gitTutorial2 src Person
Project
gitTutorial2 [gitTutorial] ~/D
.idea
src
Instructor
Main
Person
Student
gitTutorial.iml
External Libraries
Person.java
Person
1 /**
2 * Created by ethan on 2017-02-07.
3 */
4 abstract class Person {
5 String name;
6 String emailAddress;
7
8 Person(String name, String emailAddress){
9 this.name=name;
10 this.emailAddress=emailAddress;
11 }
12 }
13
Version Control: Local Changes Log Console
Branch: All User: All Date: All Paths: All
Initial commit origin & master ethan 07/02/17 12:23 PM
Commit details
Git Branches
New Branch
Checkout Tag or Revision...
Remote Branches
origin/master
Current branch: master
9: Version Control Terminal 6: TODO
Create and checkout new branch
4:16 LF UT Git: master
1: Project 7: Structure 2: Favorites
Database Maven Projects Ant Build


Create New Branch
New branch name:
dev
OK
Cancel


gitTutorial2 - [~/Desktop/gitTutorial2] - [gitTutorial] - ~/Desktop/gitTutorial2/src/Person.java - IntelliJ IDEA 2016.3.4
File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help
gitTutorial2 src Person
Project
gitTutorial2 [gitTutorial] ~/D
.idea
src
Instructor
Main
Person
Student
gitTutorial.iml
External Libraries
Person.java
Person
1 /**
2 * Created by ethan on 2017-02-07.
3 */
4 abstract class Person {
5 String name;
6 String emailAddress;
7
8 Person(String name
9 this.name=name
10 this.emailAddr
11 }
12 }
13
Git Branches
New Branch
Checkout Tag or Revision...
Local Branches
master → origin/master
Remote Branches
origin/master
Current branch: dev
Version Control: Local Changes Log Console
Branch: All User: All Date: All Paths: All
Initial commit origin & master ethan 07/02/17 12:23 PM
Commit details
Notice that current branch was automatically changed to dev.
1: Project
7: Structure
2: Favorites
Database
Maven Projects
Ant Build
9: Version Control
Terminal
6: TODO
Event Log
Branch dev was created (a minute ago)
4:16 LF UTF-8 Git: dev


gitTutorial2 - [~/Desktop/gitTutorial2] - [gitTutorial] - ~/Desktop/gitTutorial2/src/Person.java - IntelliJ IDEA 2016.3.4
File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help
gitTutorial2 src Person
Project
gitTutorial2 [gitTutorial] ~/D
.idea
src
Instructor
Main
Person
Student
gitTutorial.iml
External Libraries
Person.java
Person
/**
* Created by ethan on 2017-02-07.
abstract class Person {
String name;
String emailAddress;
Person(String name,
this.name=name;
this.emailAddre
}
}
Git Branches
New Branch
Checkout Tag or Revision...
Local Branches
dev
feature1
feature2
Remote Branches
origin/master
Current branch: master
Create two new branches for feature1 and feature2.
Checkout to feature1.
Commit details
1: Project
7: Structure
2: Favorites
Database
Maven Projects
Ant Build
9: Version Control
Terminal
6: TODO
Event Log
Create and checkout new branch
4:16 LF UTF-8 Git: master

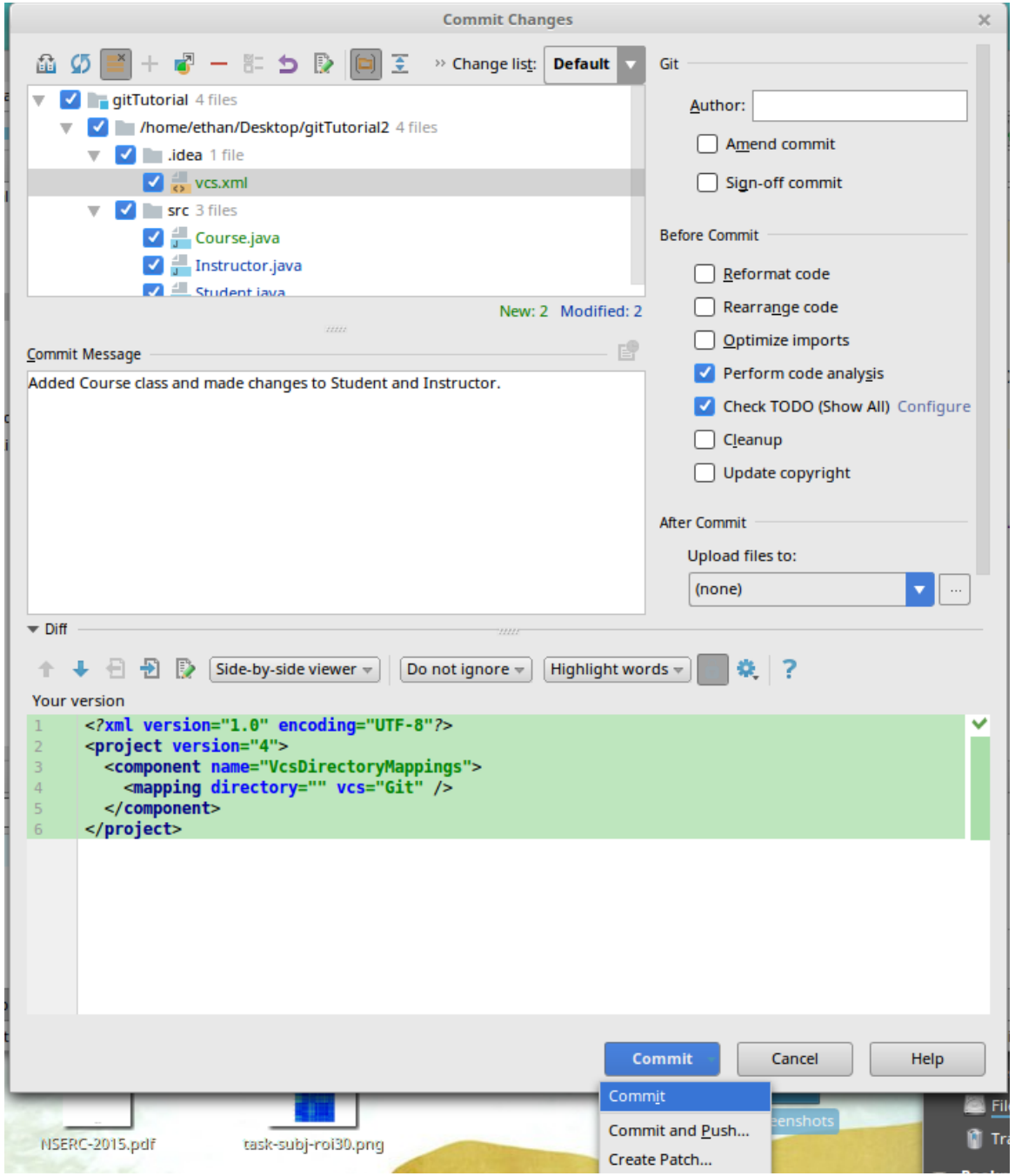

gitTutorial2 - [~/Desktop/gitTutorial2] - [gitTutorial] - ~/Desktop/gitTutorial2/src/Course.java - IntelliJ IDEA 2016.3.4
File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help
gitTutorial2 src Course
Project
Person.java
Course.java
Student.java
gitTutorial2 [gitTutorial] ~/D
.idea
src
Course
Instructor
Main
Person
Student
gitTutorial.iml
External Libraries
Course Course()
1 /**
2 * Created by ethan on 07/02/17.
3 */
4 public class Course {
5 String cname;
6 String description;
7
8 public Course(String cname, String description) {
9 this.cname = cname;
10 this.description = description;
11 }
12 }
13
Version Control: Local Changes Log Console
Branch: All User: All Date: All Paths: All
Initial commit origin & master ethan 07/02/17 12:23 PM
Commit details
Create a new class Course as above. Make sure the current branch is feature1.
9: Version Control Terminal 6: TODO Event Log
Checked out feature1 (2 minutes ago)
8:54 LF UTF-8 Git: feature1
1: Project 7: Structure 2: Favorites
Database Maven Projects Ant Build

Modify Student class to have:

- Instance variable: List<Course> courseList
- Method: public void addCourse(Course c)

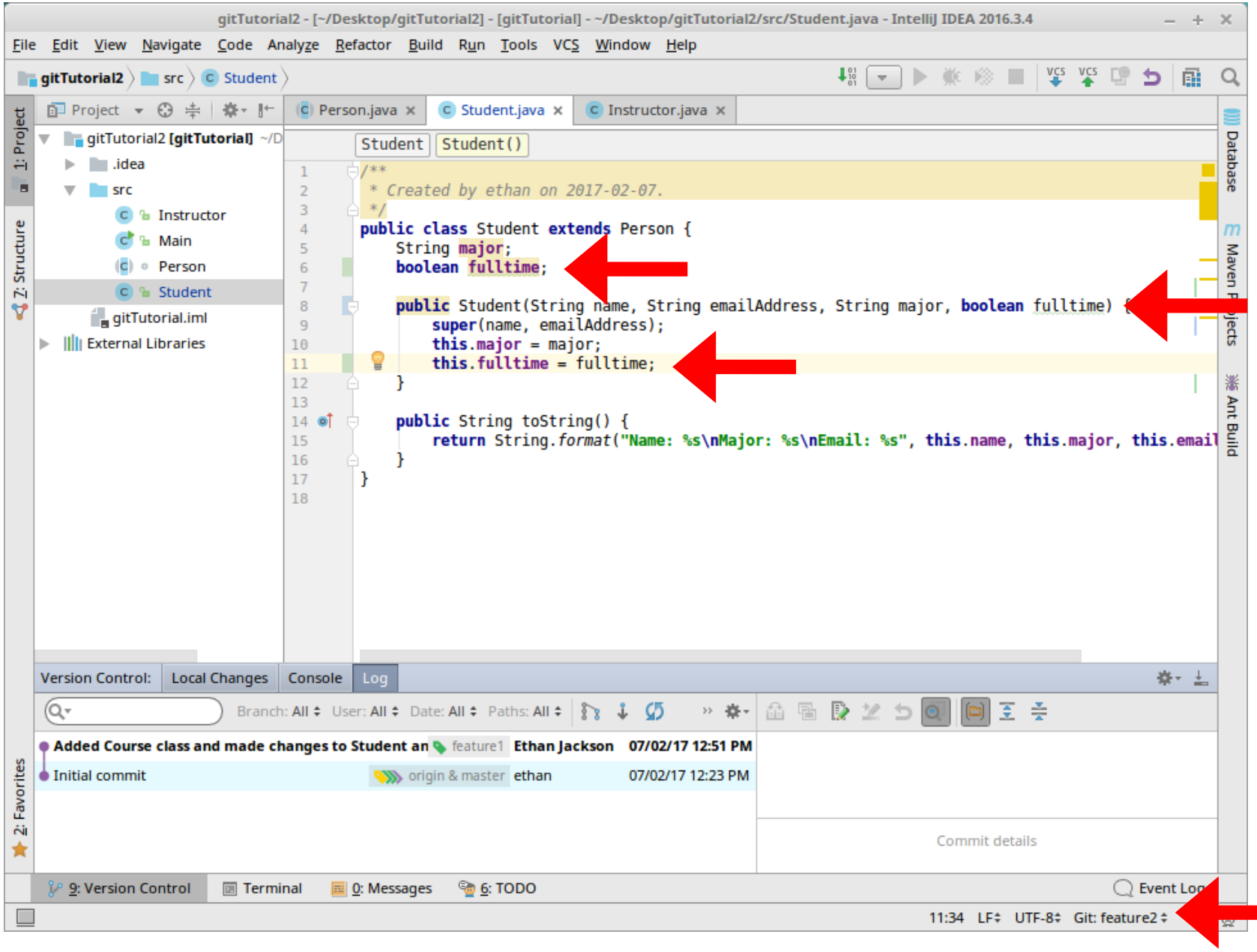
Modify Instructor class to have:

- Instance variable Course favCourse
- Constructor parameter favCourse

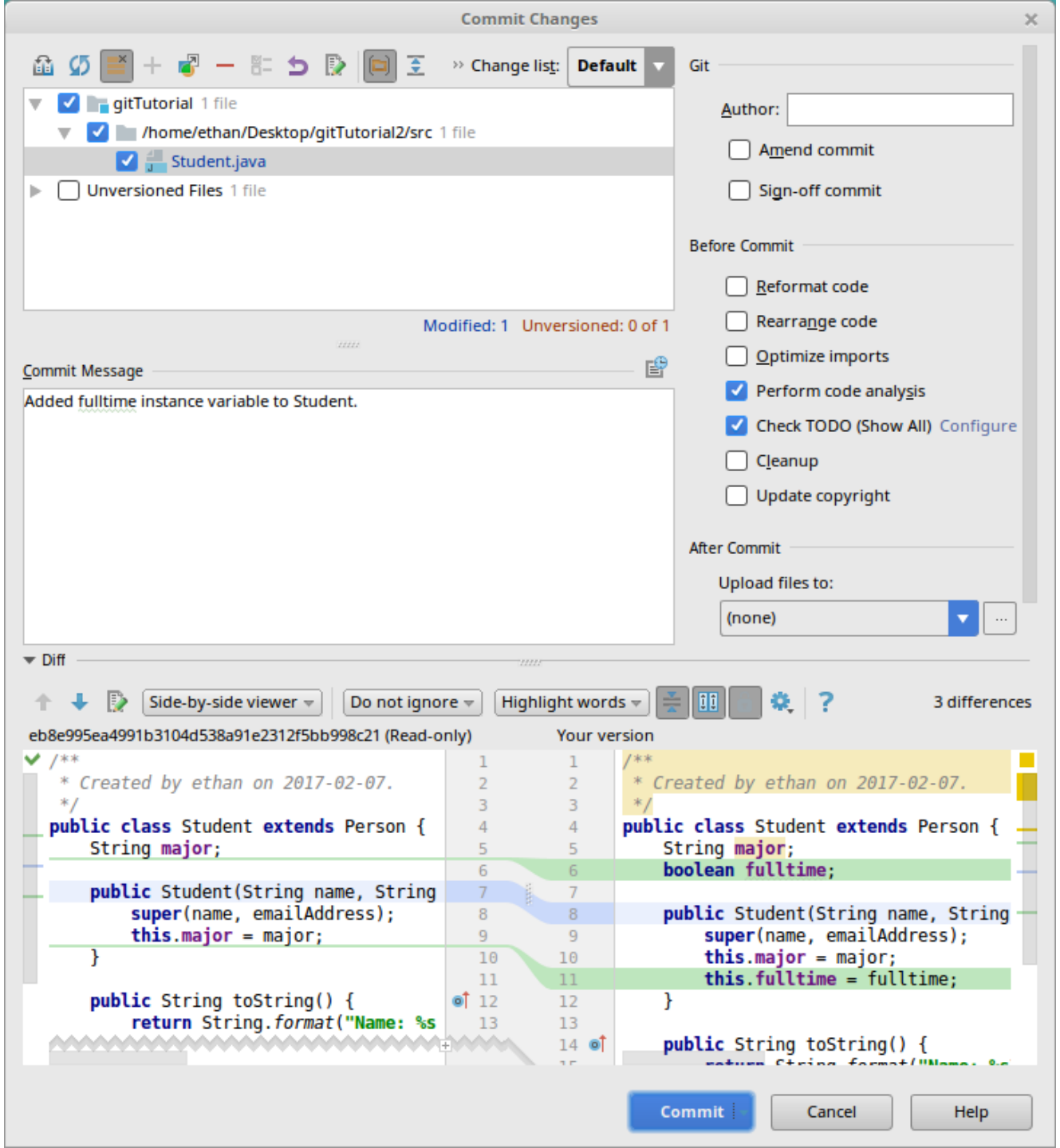


Commit your changes.

Checkout to **feature2** and modify Student:



Make sure you are working in the **feature2** branch.



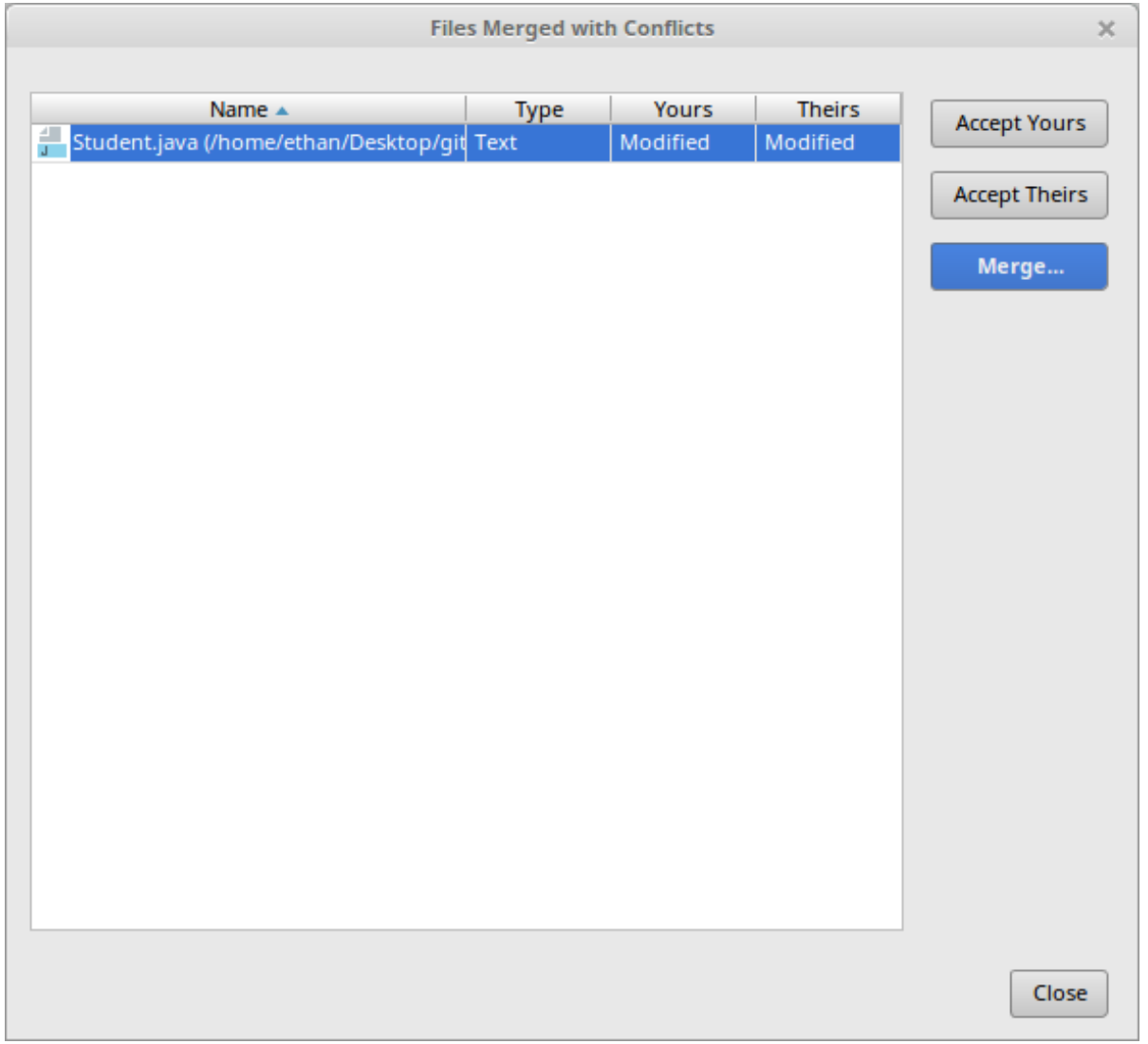
Commit your changes to the **feature2** branch.

Checkout to **dev**.

Merge **feature1**.

Then merge **feature2.**
→ **conflicts**

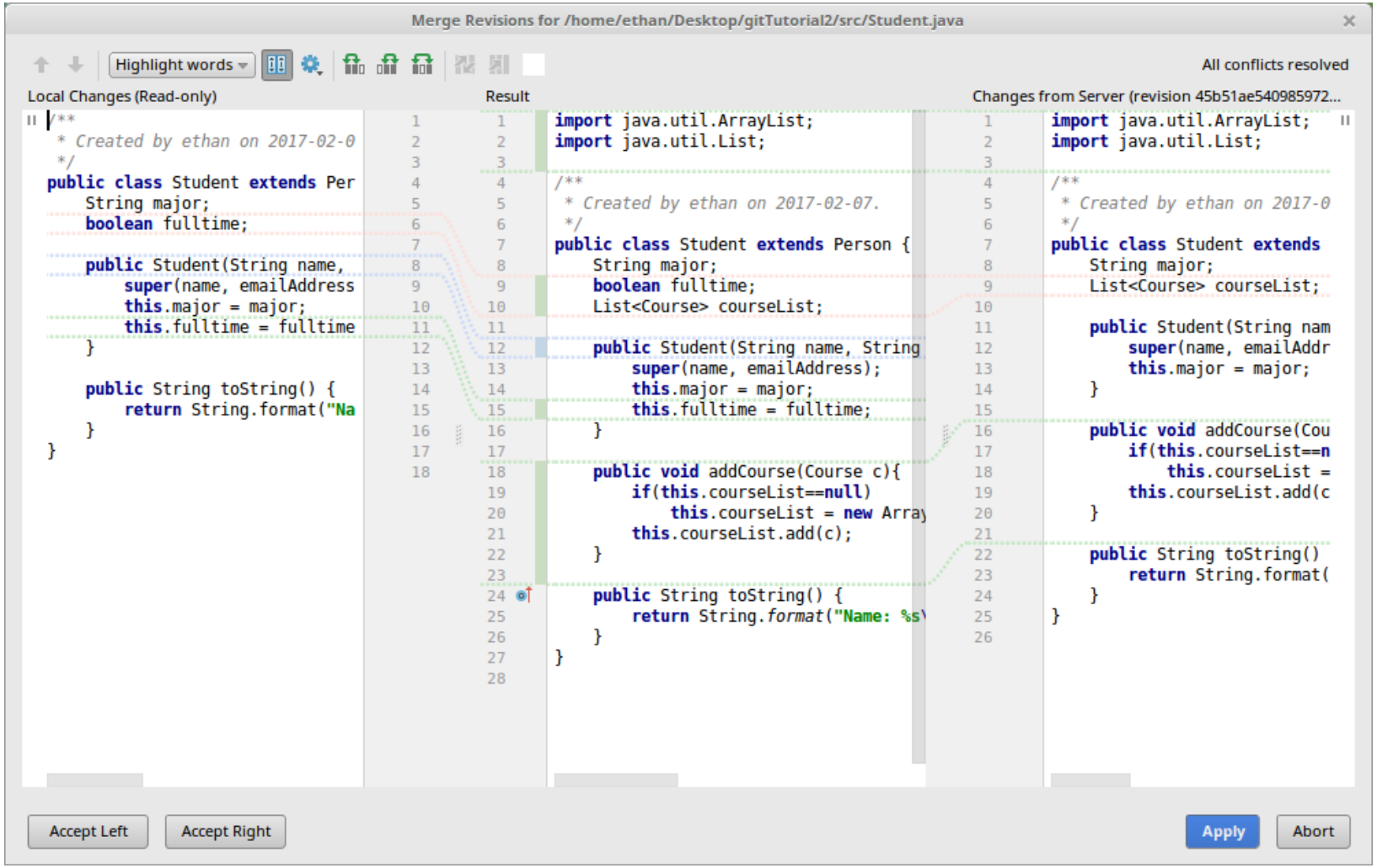
Since both branches contained changes to Student, we need to **merge** the conflicts.



Use the merge tool to incorporate the changes from both **feature1** and **feature2**.

This needs to be done **carefully** and **thoughtfully**!

IntelliJ provides suggestions but the conflict resolution is up to you.



Tutorial Submission

Now the **dev** branch contains the changes from both **feature1** and **feature2**.

You can now checkout to **master** and merge it with **dev**.

Finally, **commit and push** the updated **master** branch to your GitHub account. You must submit the URL to your updated repository .git file via OWL by **Midnight on Wednesday Feb 15th**

You must **not make any changes after the due date**.

Tutorial Submission

Example submission:

https://github.com/ethamajin/gitTutorial.git

In other words, do not submit anything other than your repository's .git URL in the text submission area. **If you don't follow the submission instructions, you will get zero marks for the tutorial.**

Recommended git usage style

There are many ways you can choose to use git to manage the development of your project. The only **requirement** is that the **master** branch is your primary **stable** branch.

You should not merge anything to **master** until it is ready for evaluation. We will only use your **master** branch for evaluations – no exceptions!

Always do development/testing/fixes in other branches.

Recommended git usage style

The following is my recommended usage style for git.

- Use a primary development branch (**dev**)
- Each team member branches **dev** to create their own personal development branch **(myDev)**
- For each major feature, branch **myDev** into a new feature branch (**featureX)**
- When a feature is complete, merge **featureX** into **myDev**
- Merge **myDev** branches into **dev** for unstable builds
- Merge **dev** into **master** for stable releases only