

Advanced Software Development: Git Workflows

**CPSC 233: Introduction to Computer Science for Computer Science Majors II
Winter 2025**

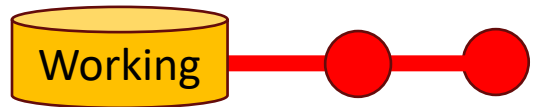
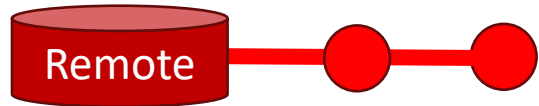
Slides via Steve Sutcliffe

Jonathan Hudson, Ph.D.
Assistant Professor (Teaching)
Department of Computer Science
University of Calgary

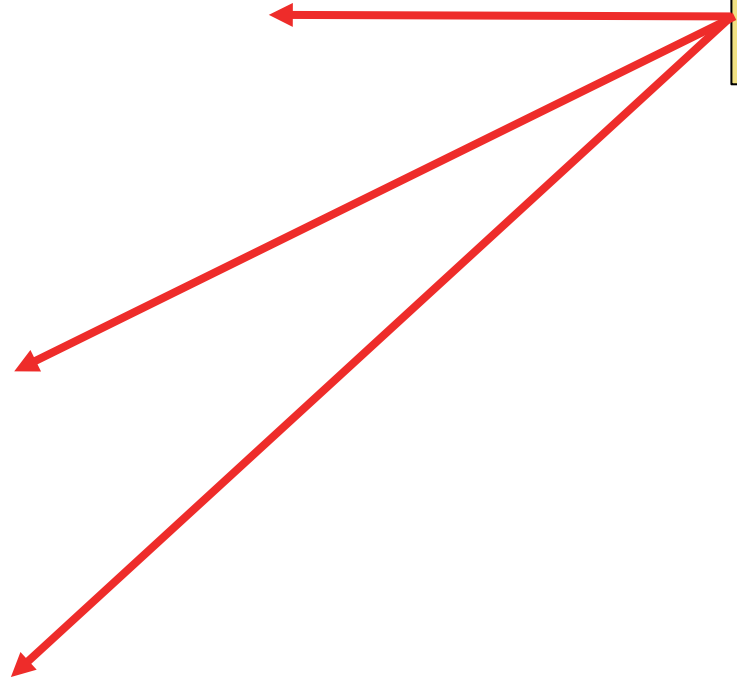
Wednesday, January 8, 2025



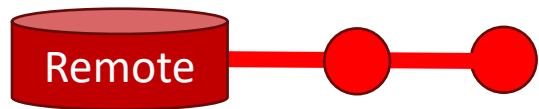
Workflow: the three realms



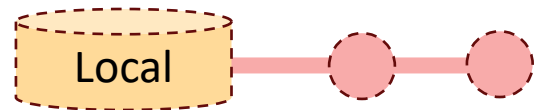
These are all copies of the same multiverse



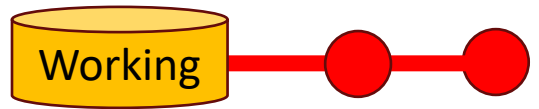
Workflow: the three realms



This is the version the world sees; all magicians have access to this one

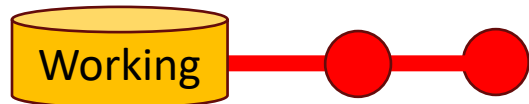
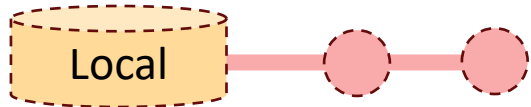
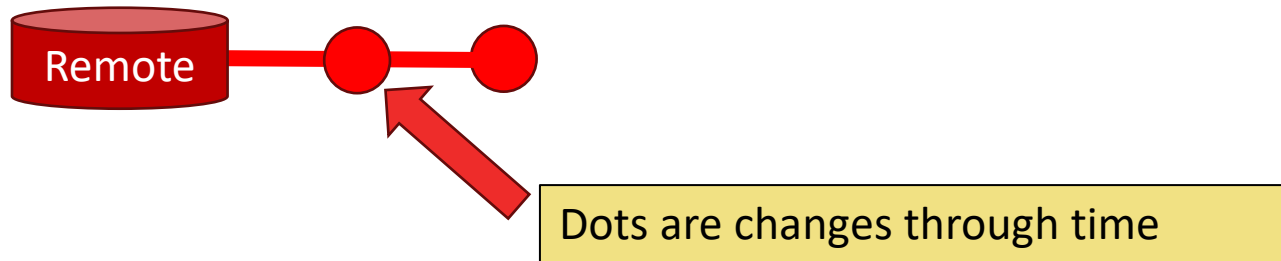


This is the version only Barry has access to, it is **git's local copy** of the "remote", it is invisible to Barry.



This is the version Barry works on

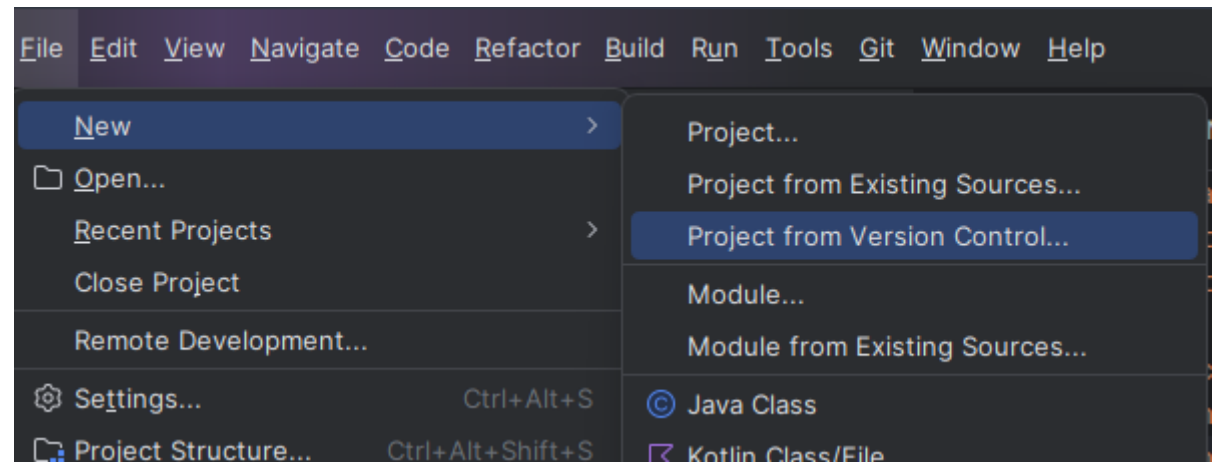
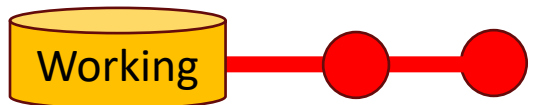
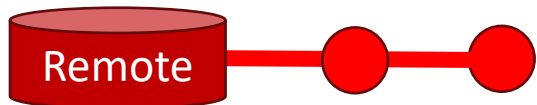
Workflow: the three realms



Solo Work

Clone

Workflow: Clone the Repo

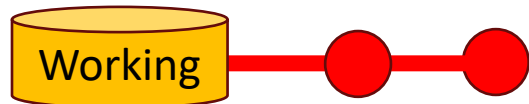
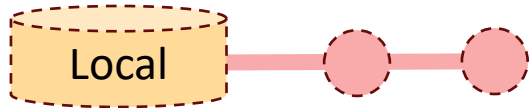
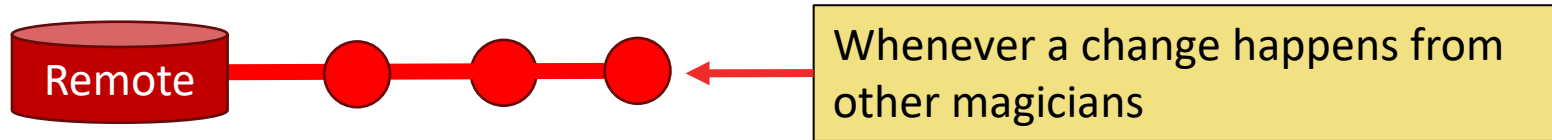


Barry creates this when he clones the repo down to his local machine

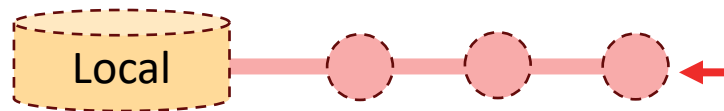
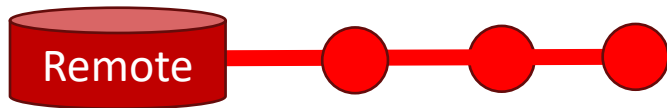


Pull

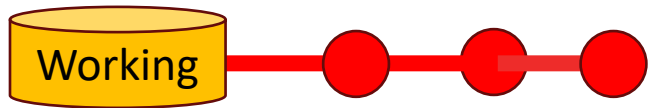
Workflow: pull (update in intellij)



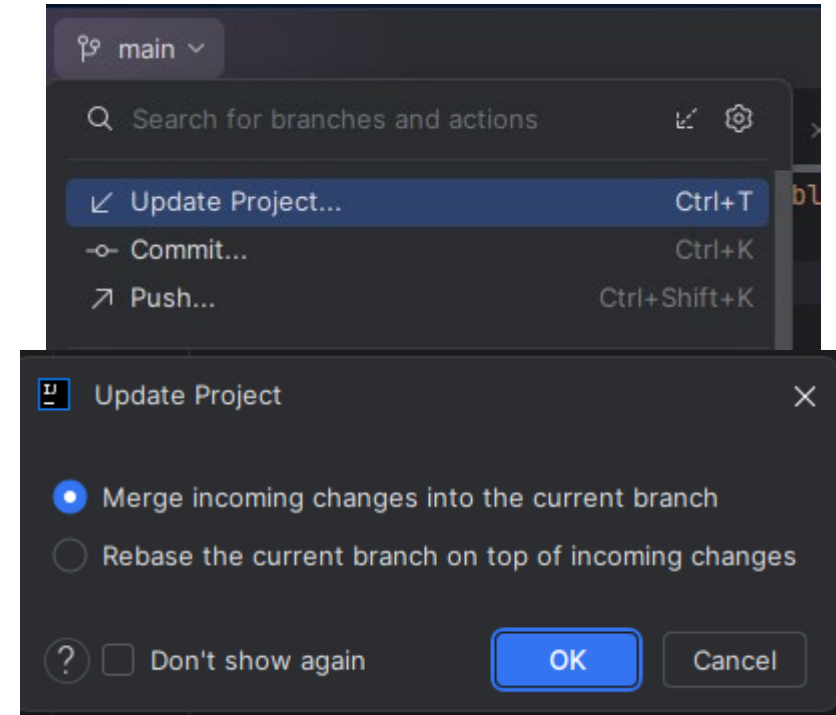
Workflow: pull (update in intellij)



Barry must update his version by pulling those changes from the remote

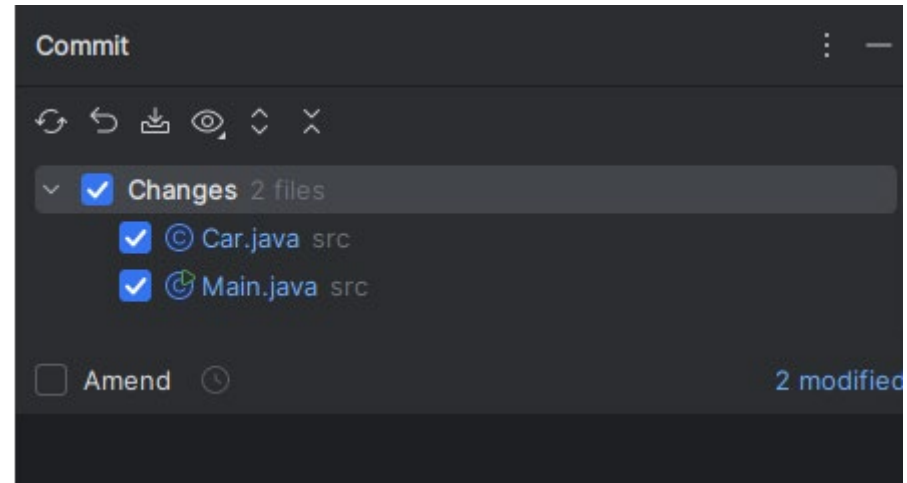
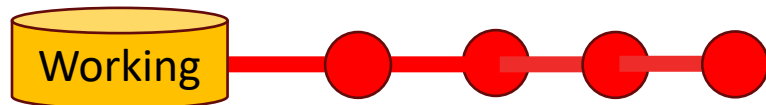
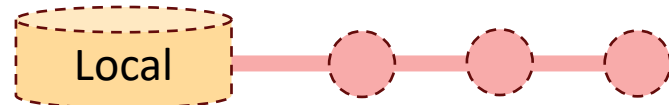
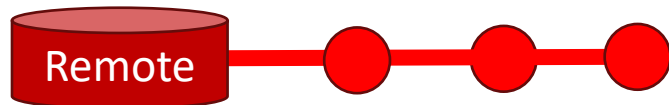


That 'pull' updates the area where Barry works.



Work

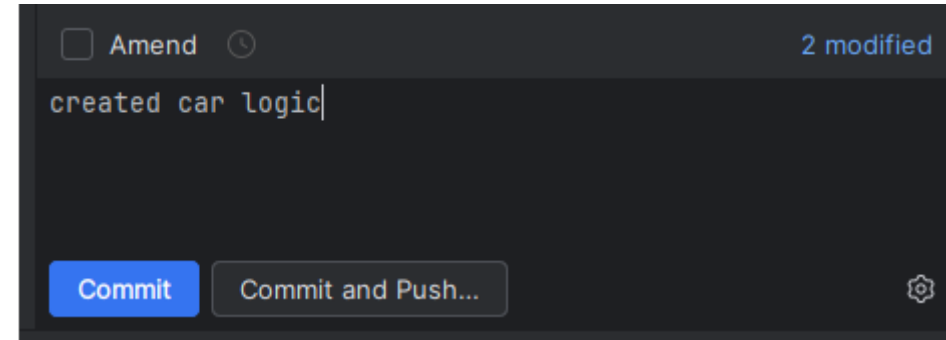
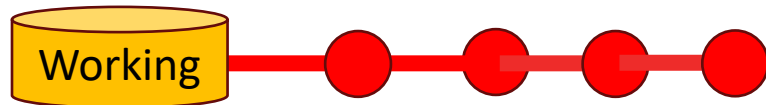
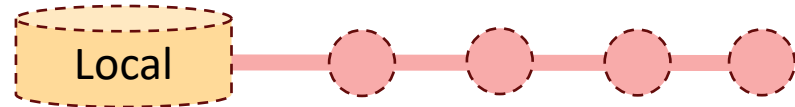
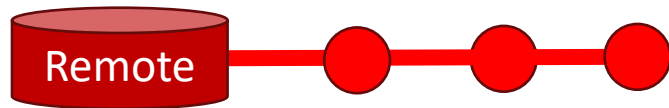
Workflow: add (checkboxes in intellij)



When Barry does something, a change in time happens only for his working

Save

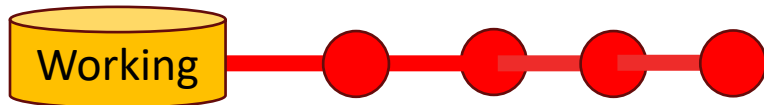
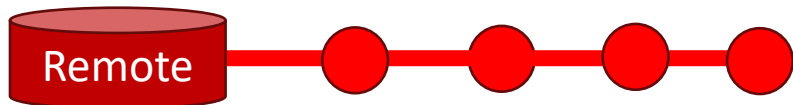
Workflow: Commit



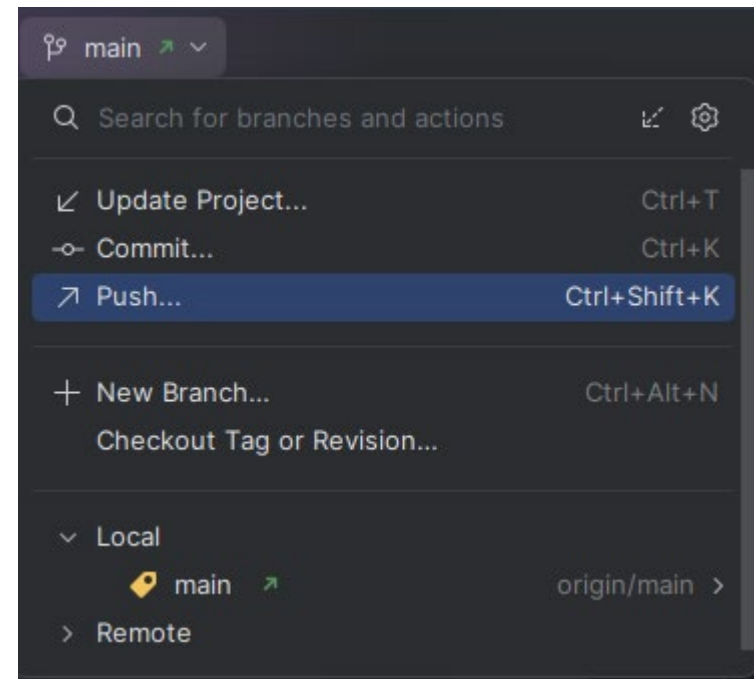
← If Barry is happy with that change, he needs to commit to it

Push

Workflow: Push



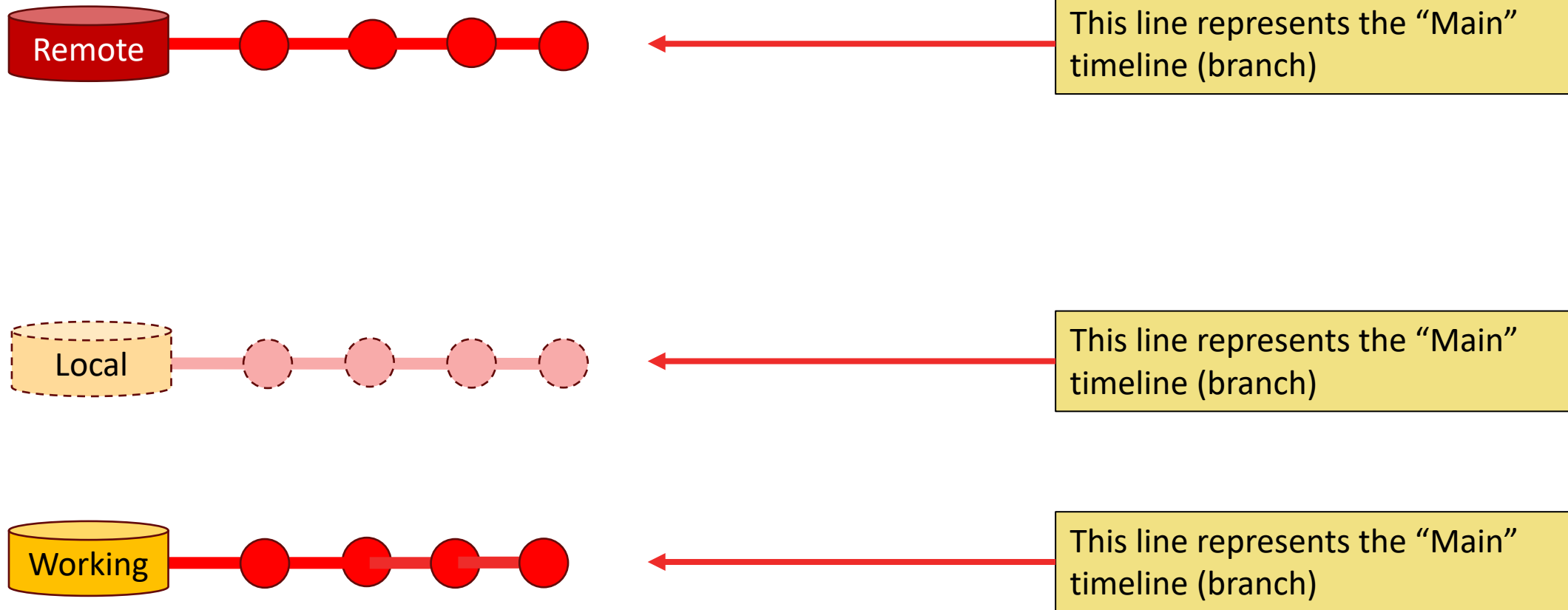
If Barry wants to share that change with other magicians, he must push those changes to the world



Branches

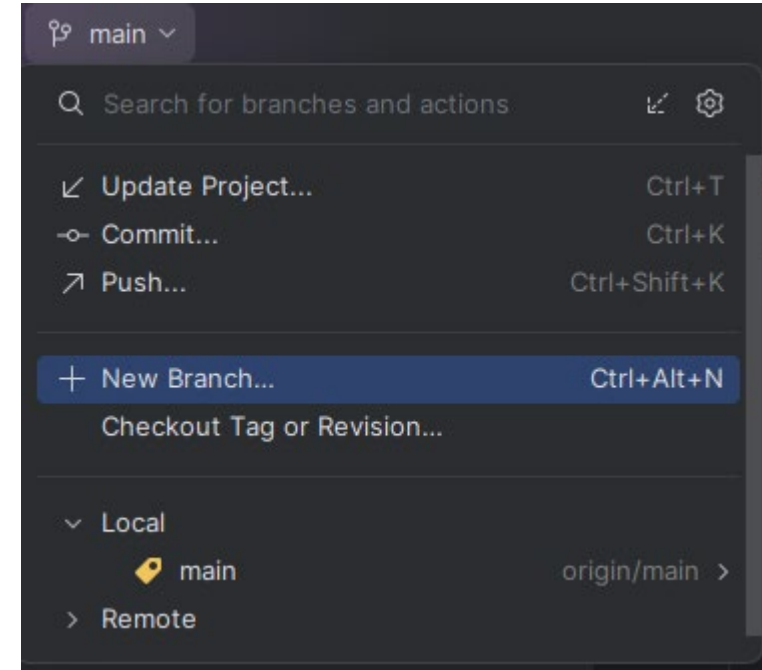
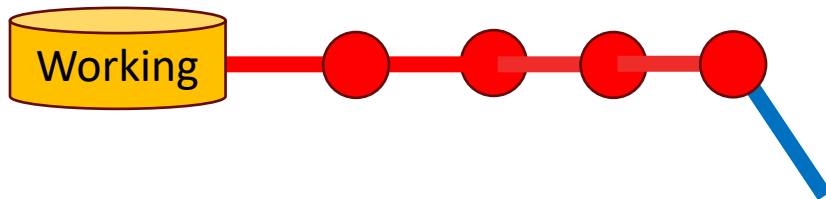
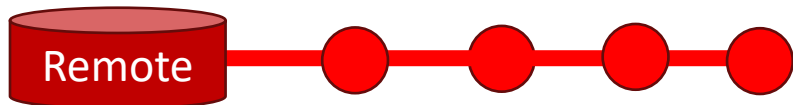
Main

Workflow: Branches -1-



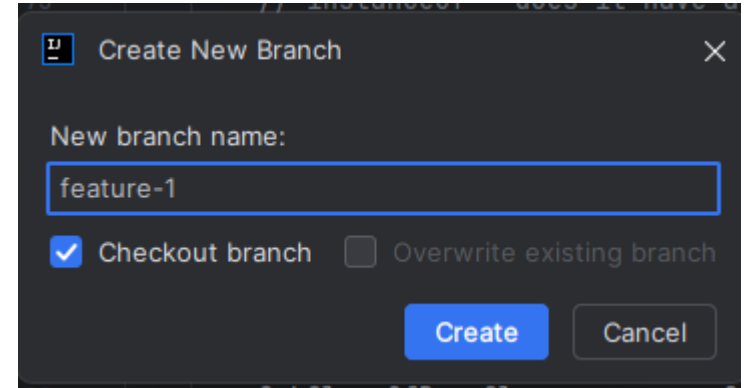
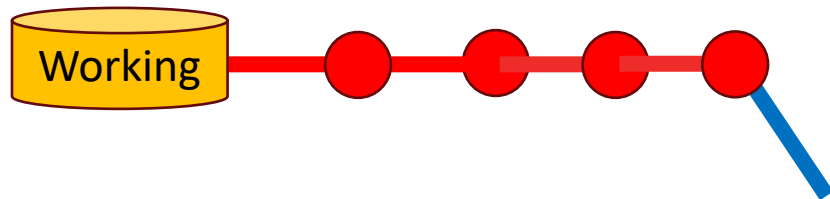
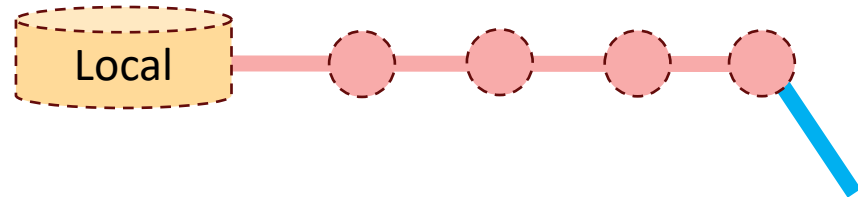
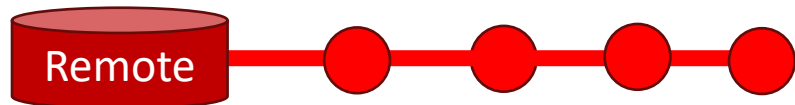
They are just different areas for working on the main timeline

Workflow: Branches -2-



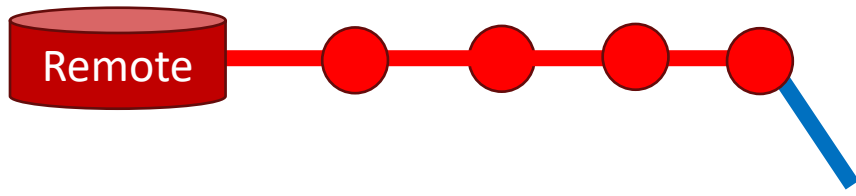
If Barry wants to experiment, he creates a new timeline (branch)

Workflow: Branches -2-

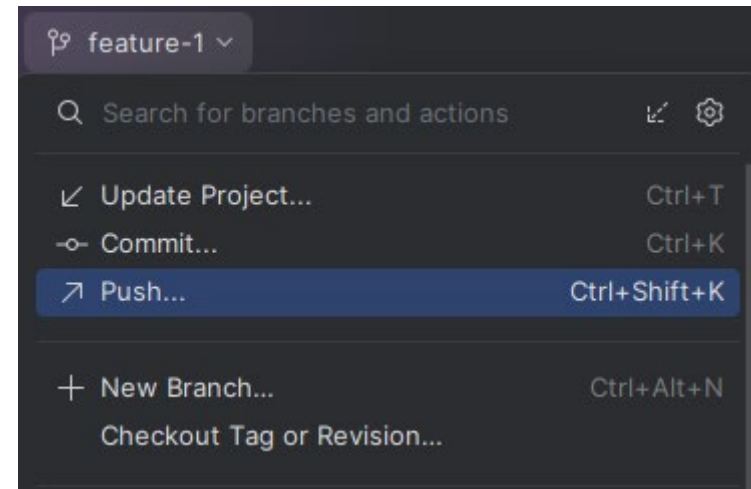
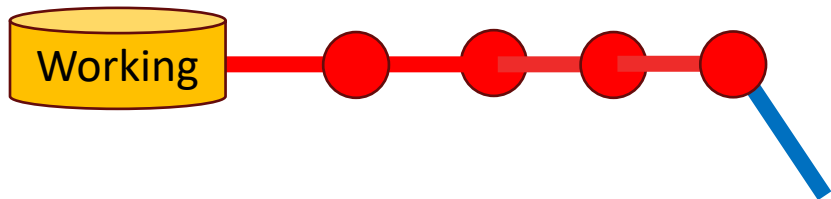
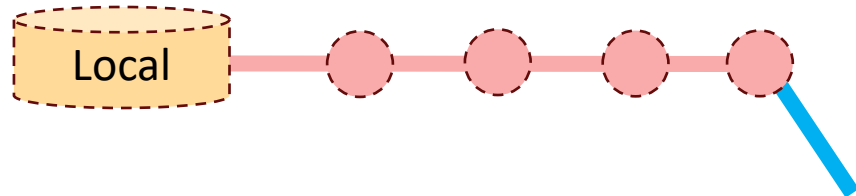


Barry's copy of the multiverse is automatically updated

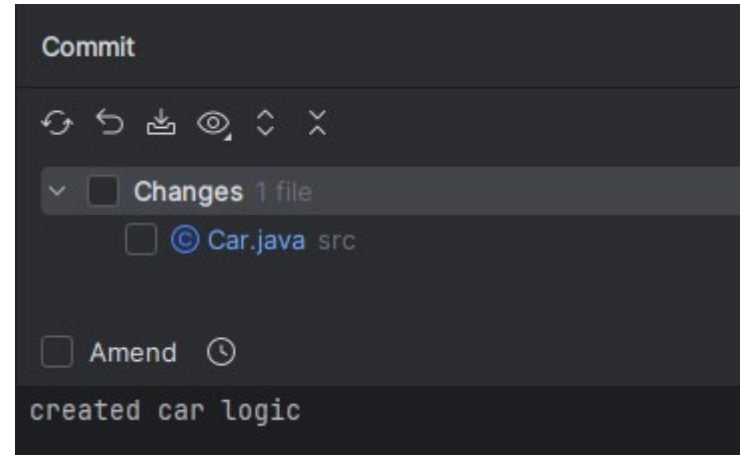
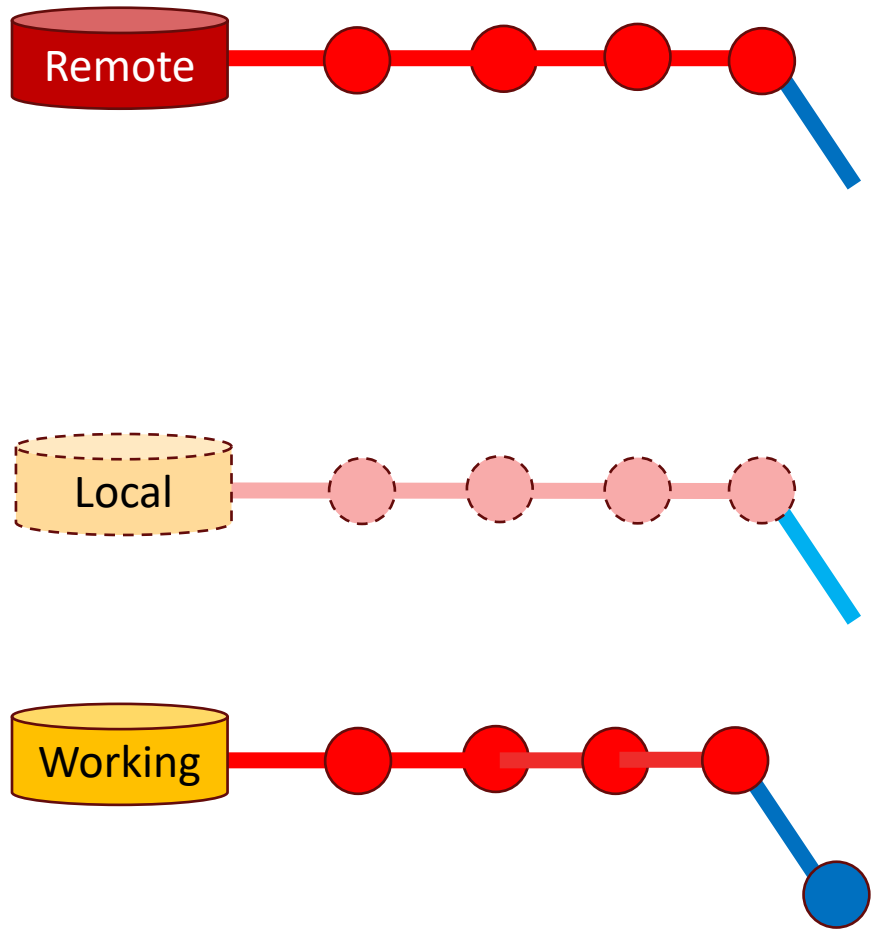
Workflow: Branches -2- (push in intellij)



But if Barry wants other magicians to see this branching timeline, he must set the upstream

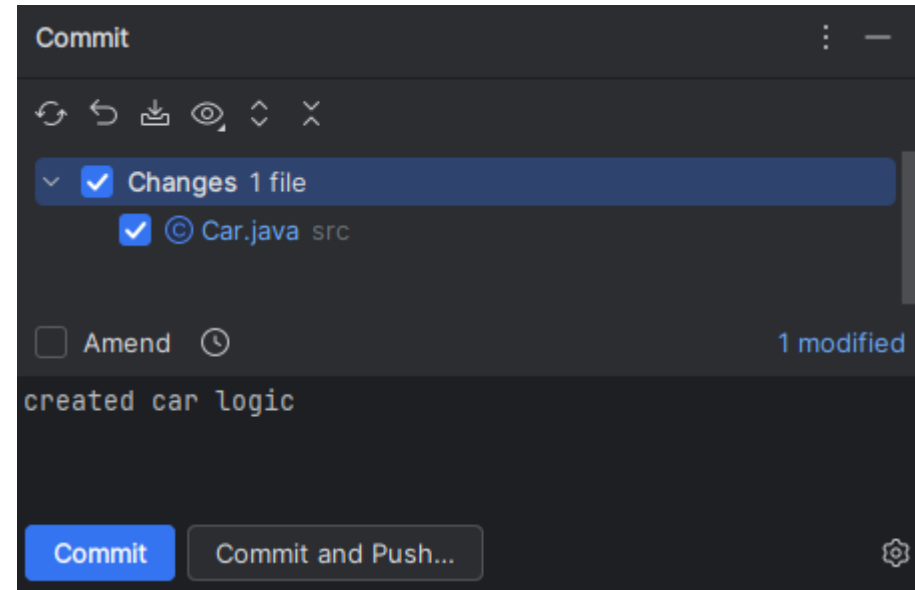
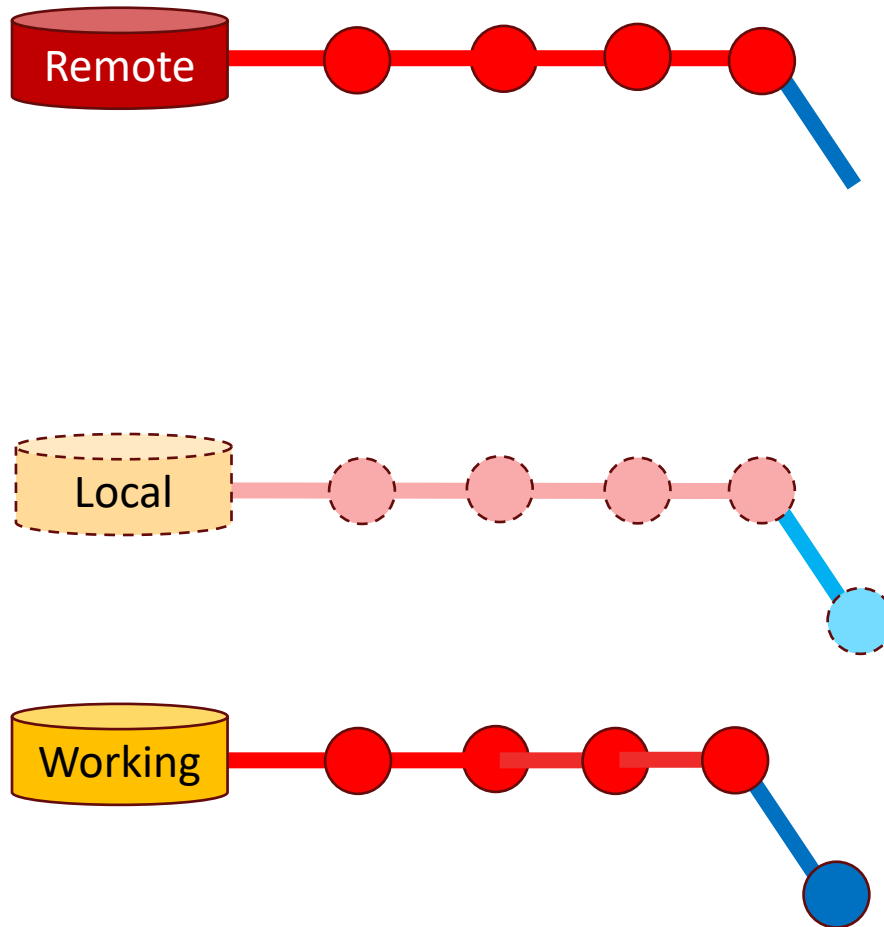


Workflow: Branches -3-



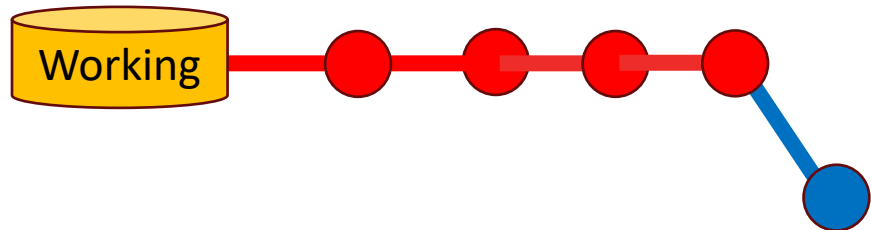
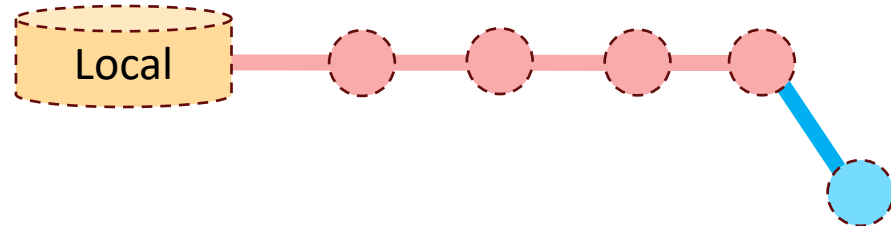
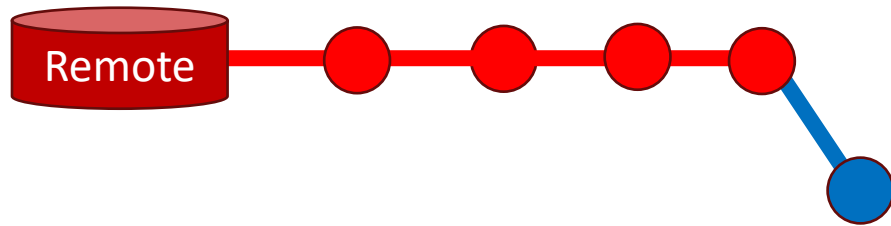
Now when Barry does something, it only effects this new branch, in the working area

Workflow: Branches -3-

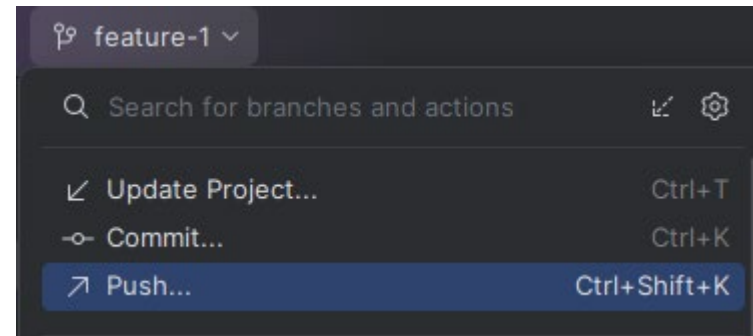


If Barry is happy with that change, he needs to "add" the changes and "commit" it.

Workflow: Branches -3-

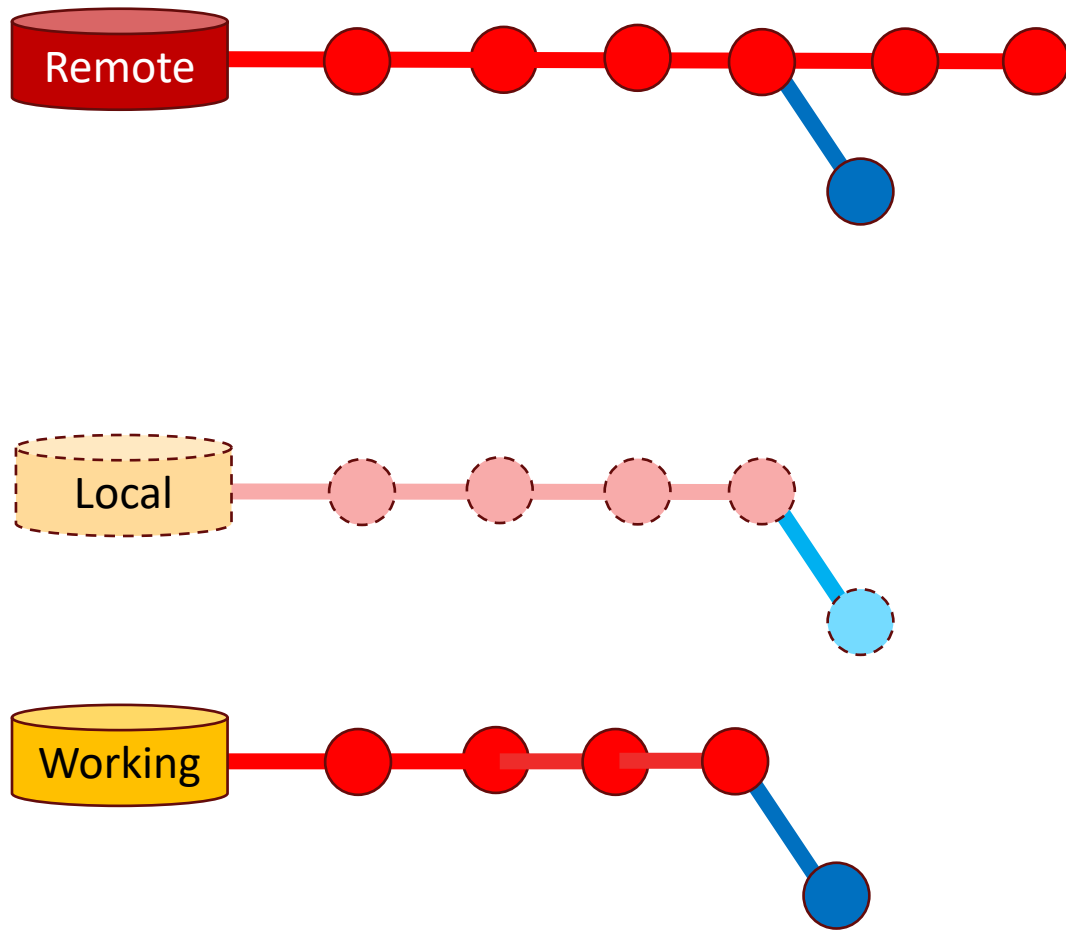


← If he wants other magicians to access it, he must "push" those changes to the remote.



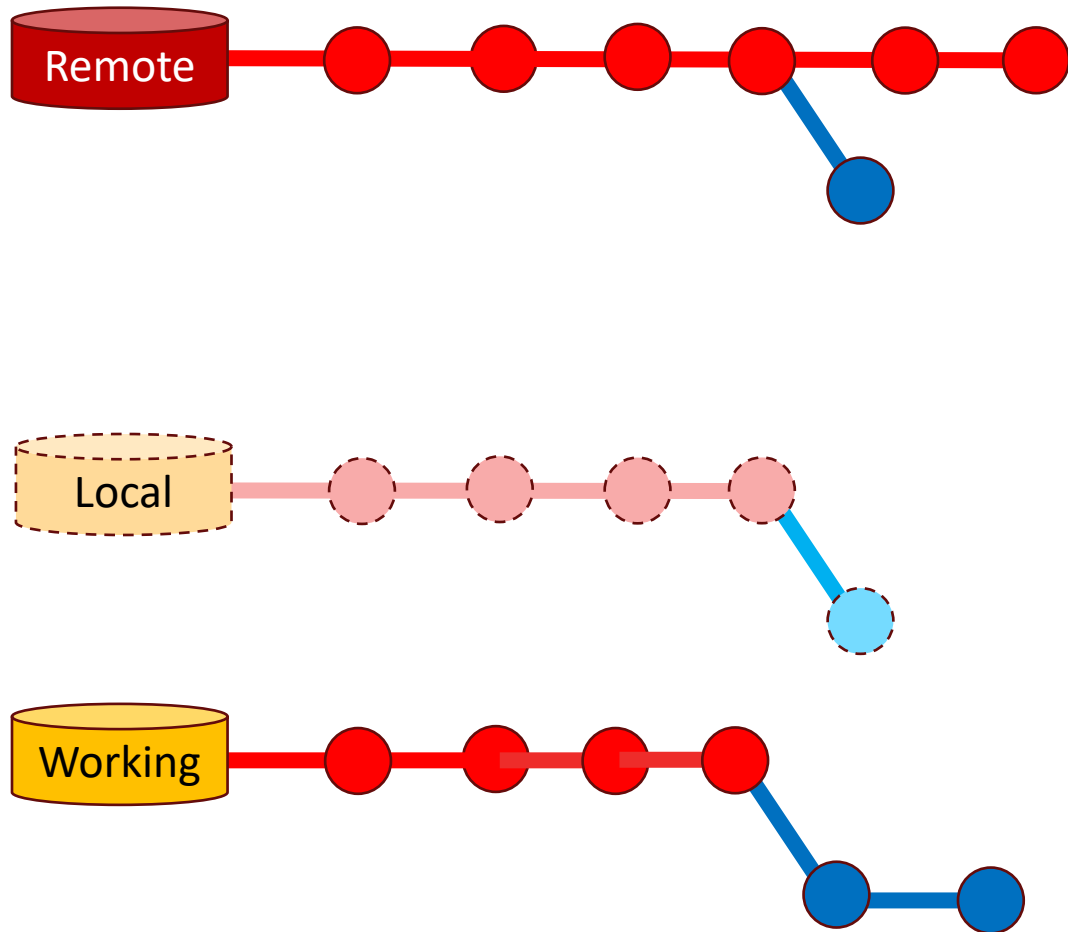
Experiment/Feature/Fix

Workflow: Merge -1-



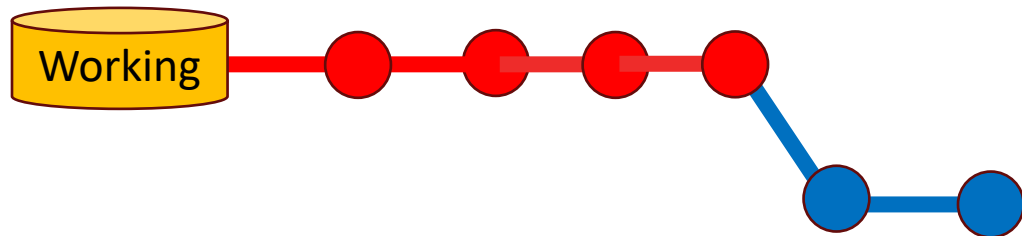
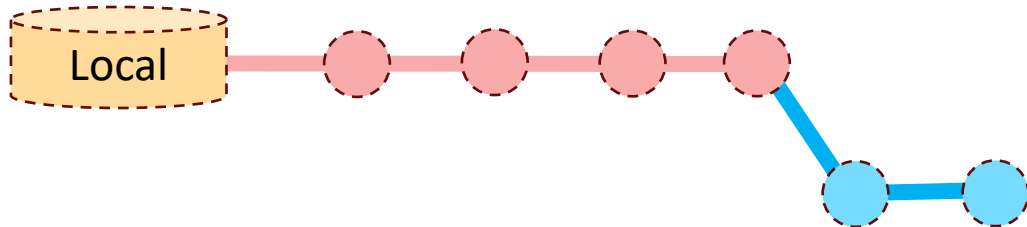
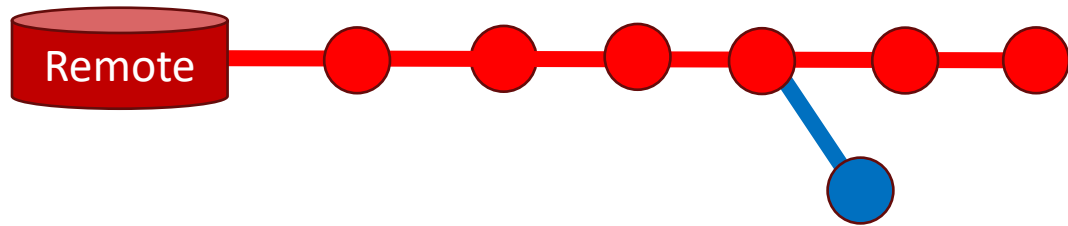
Meanwhile, other magicians are making changes for everyone to see.

Workflow: Merge -1-



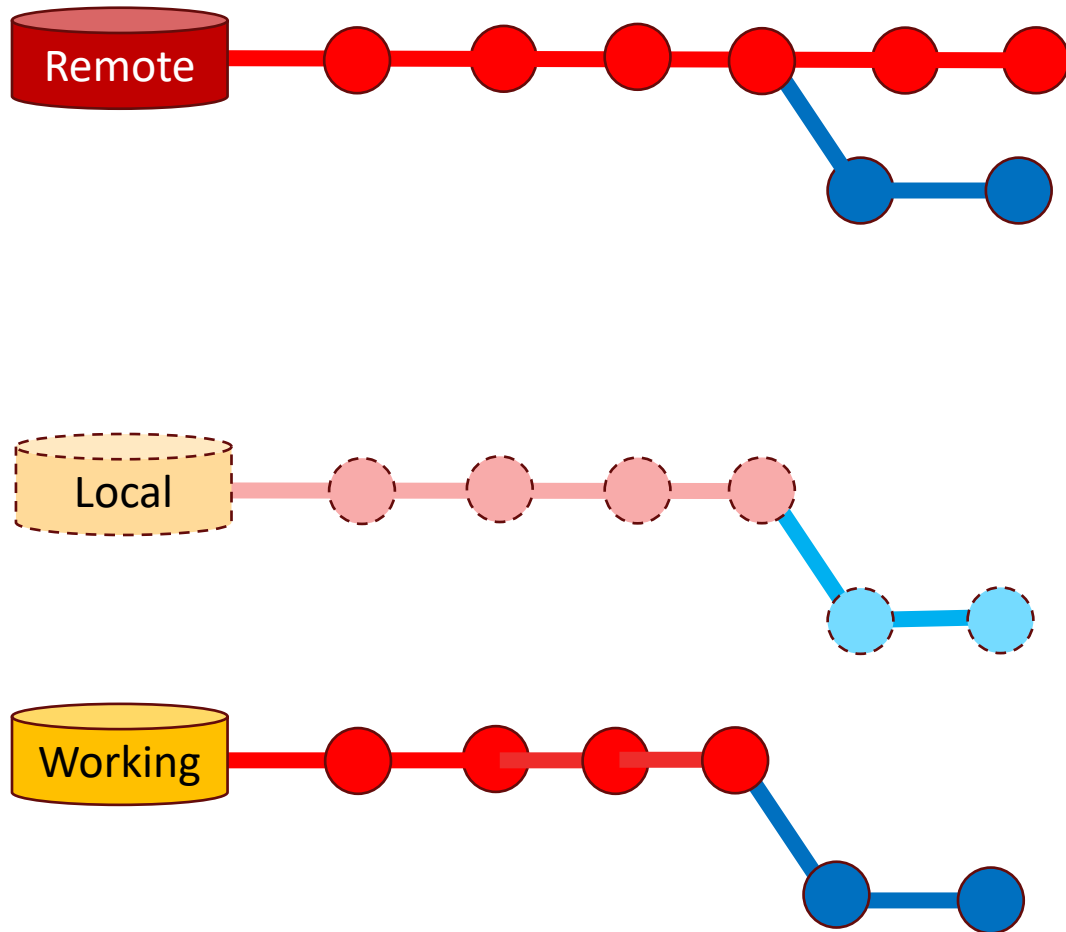
Barry can keep making his own changes.

Workflow: Merge -1-



← And committing those changes

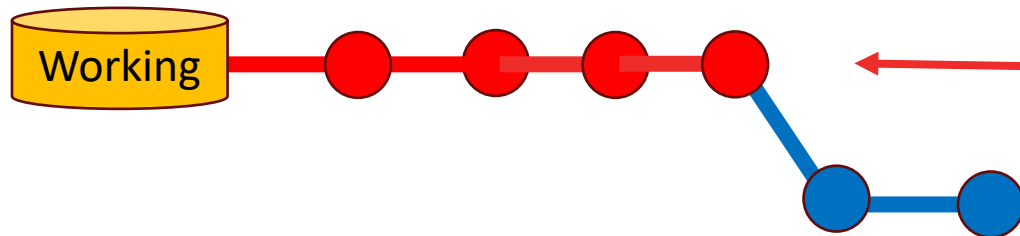
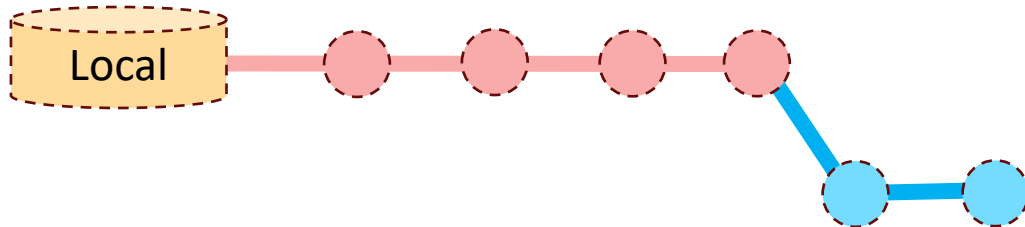
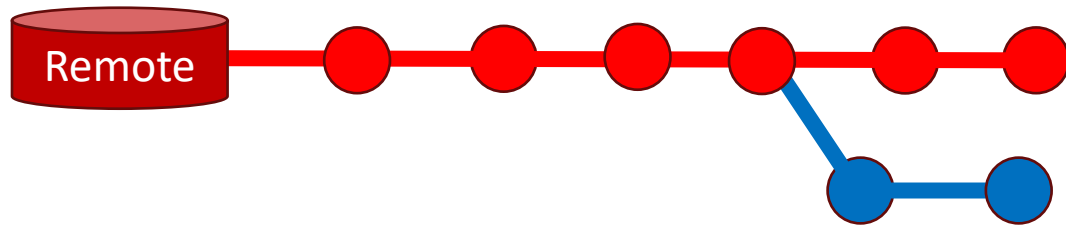
Workflow: Merge -1-



And must "push" those changes if others are to see them

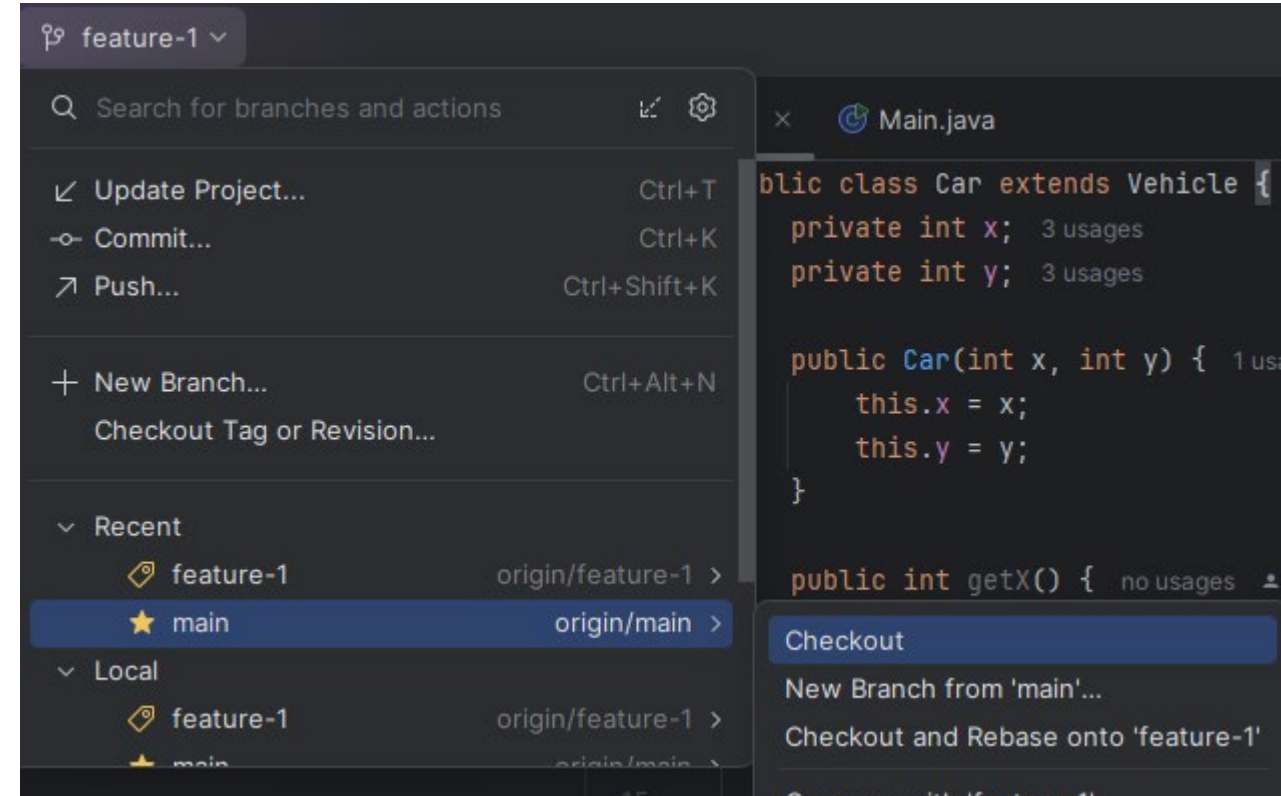
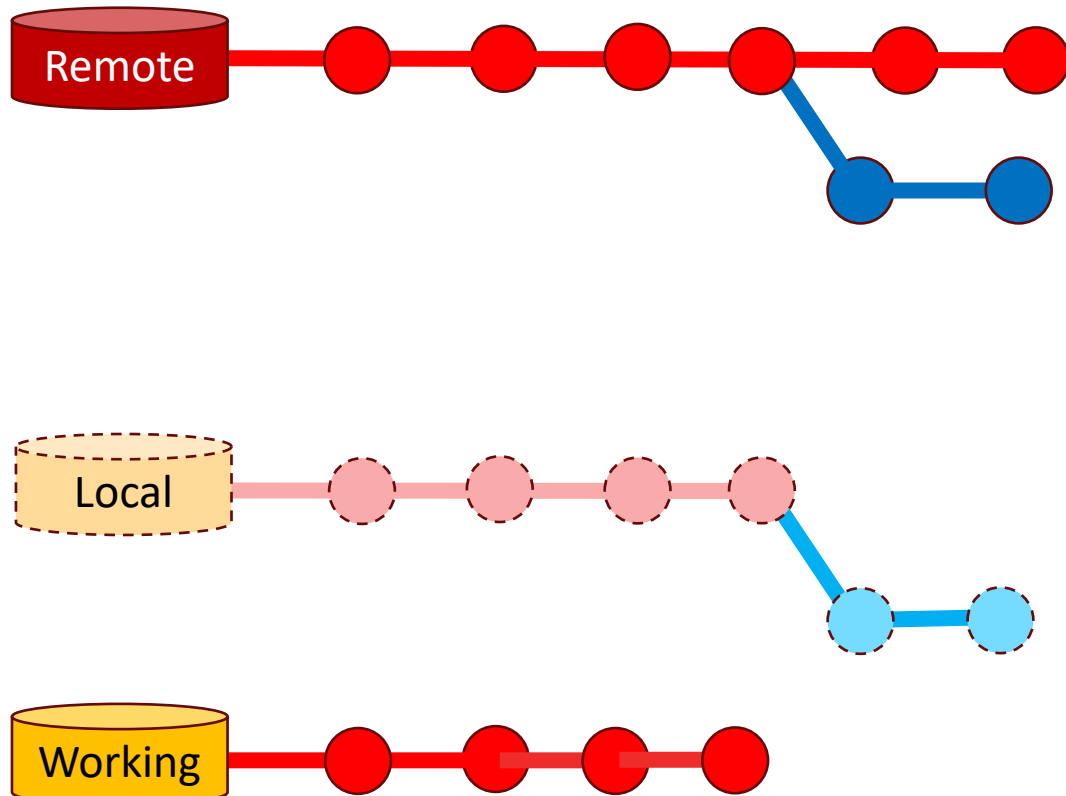
Merge

Workflow: Merge -1-



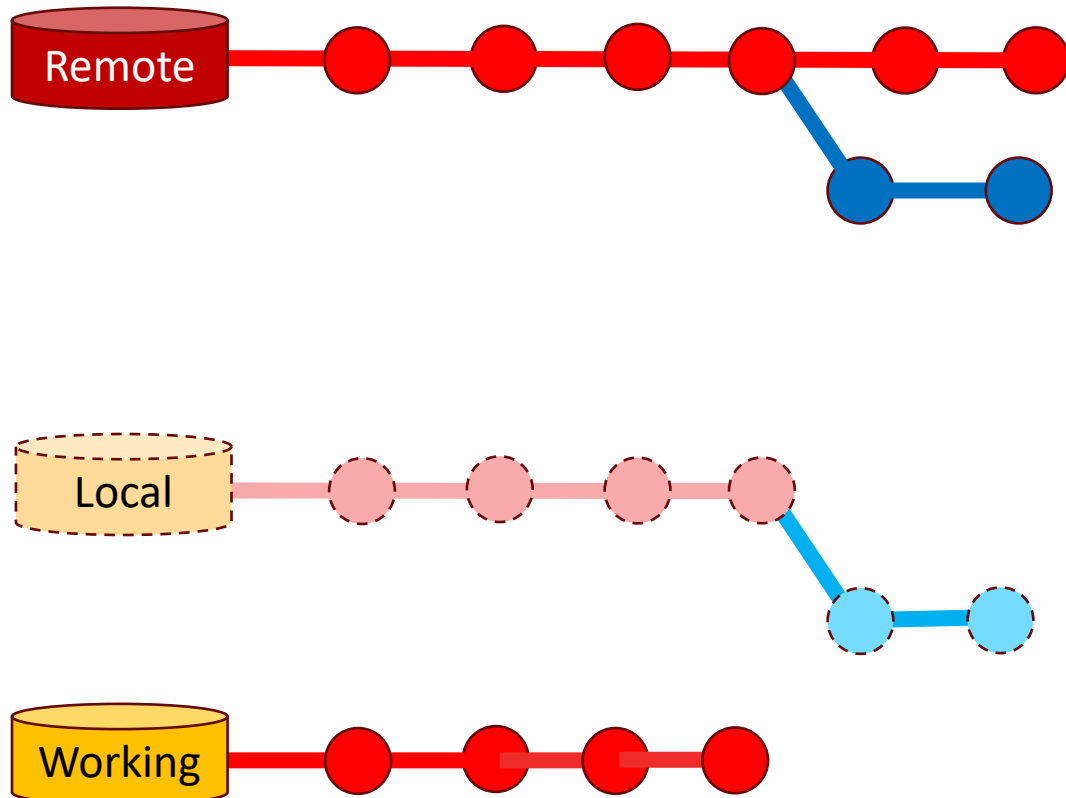
But notice, Barry's version of the main timeline is not up to date!

Workflow: Merge -1-



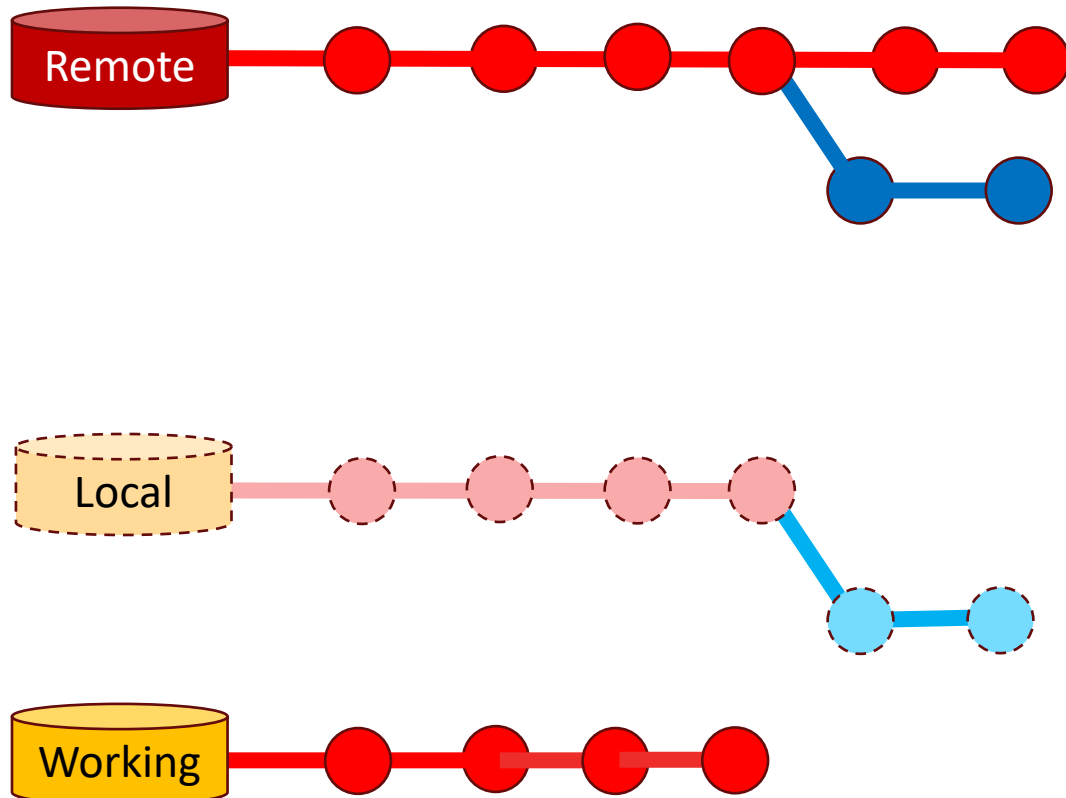
To update his version, he must "checkout" the main branch

Workflow: Merge -1-



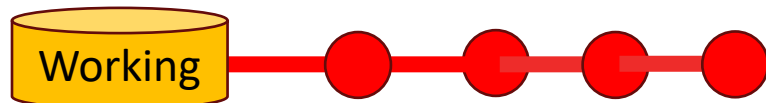
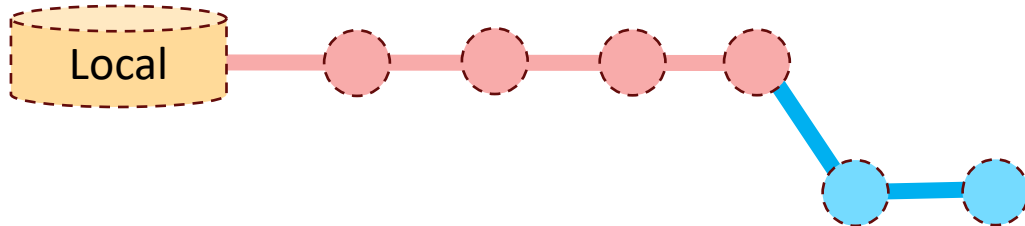
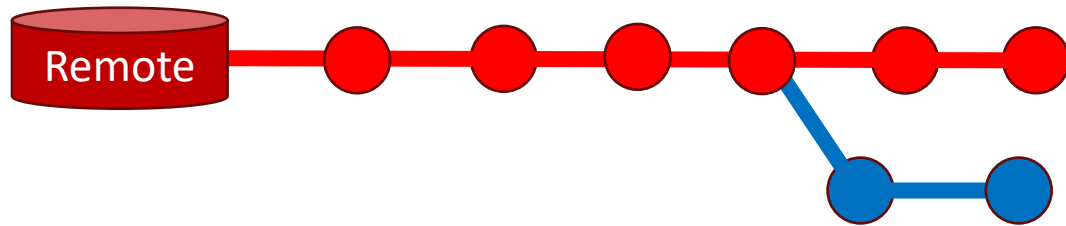
But where did all of his branch changes go?

Workflow: Merge -1-



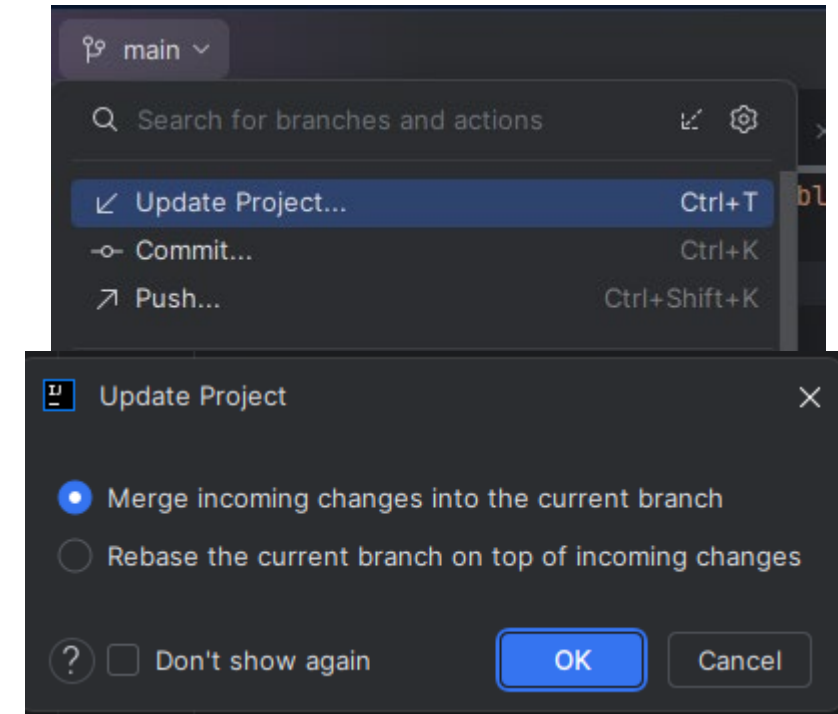
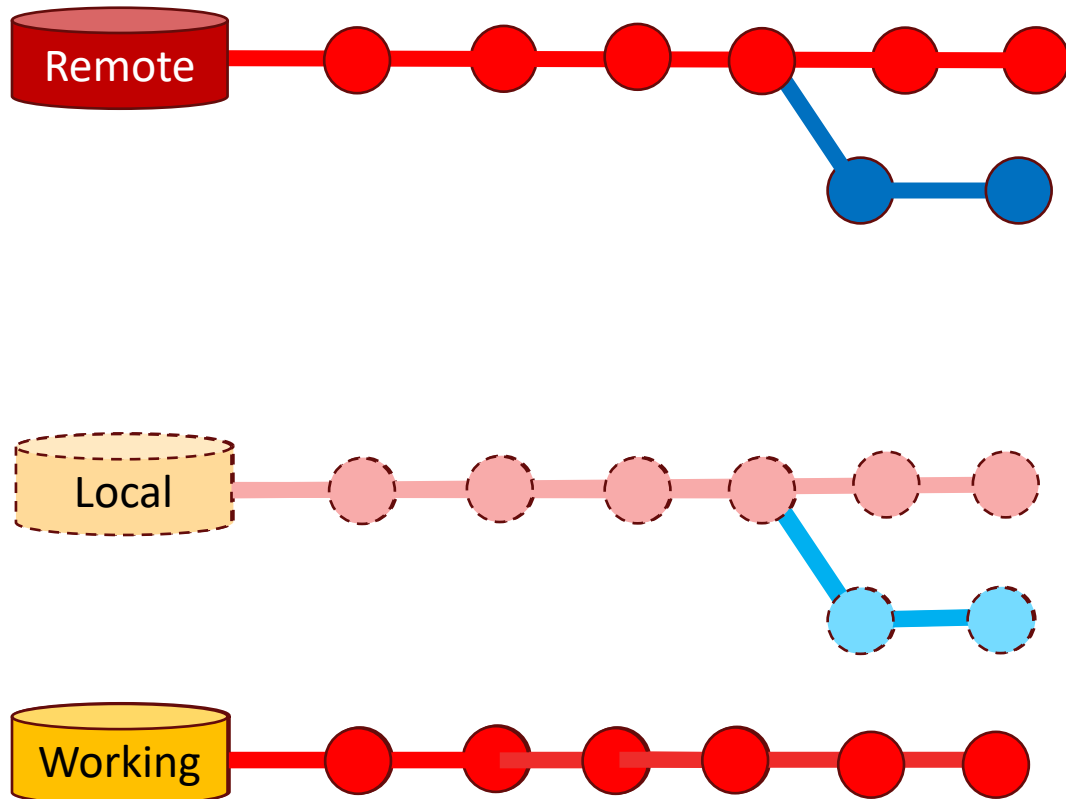
They are still preserved here, as long as he has been "committing" the changes

Workflow: Merge -1-



The working directory only shows the branch that is currently "active"

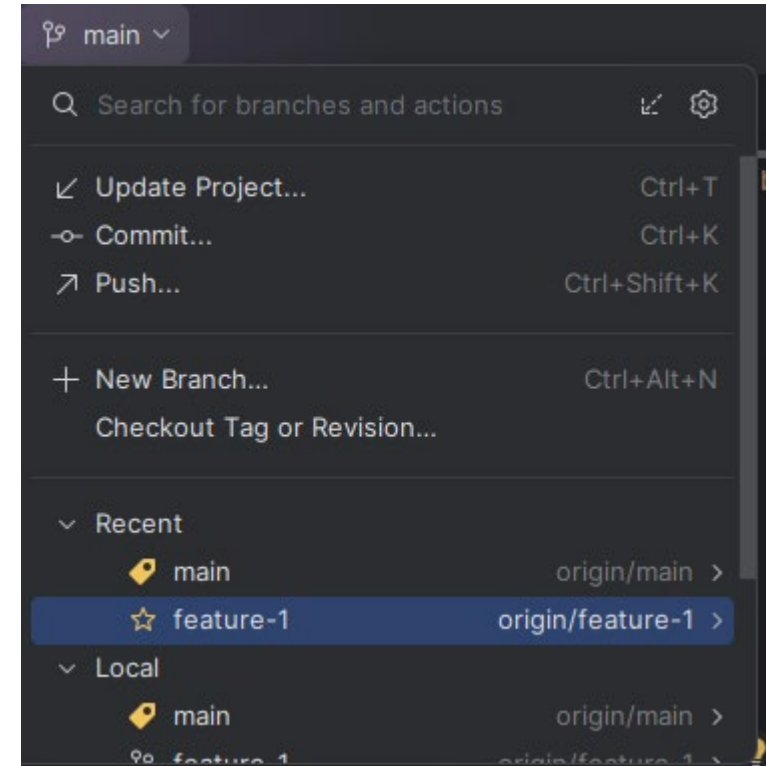
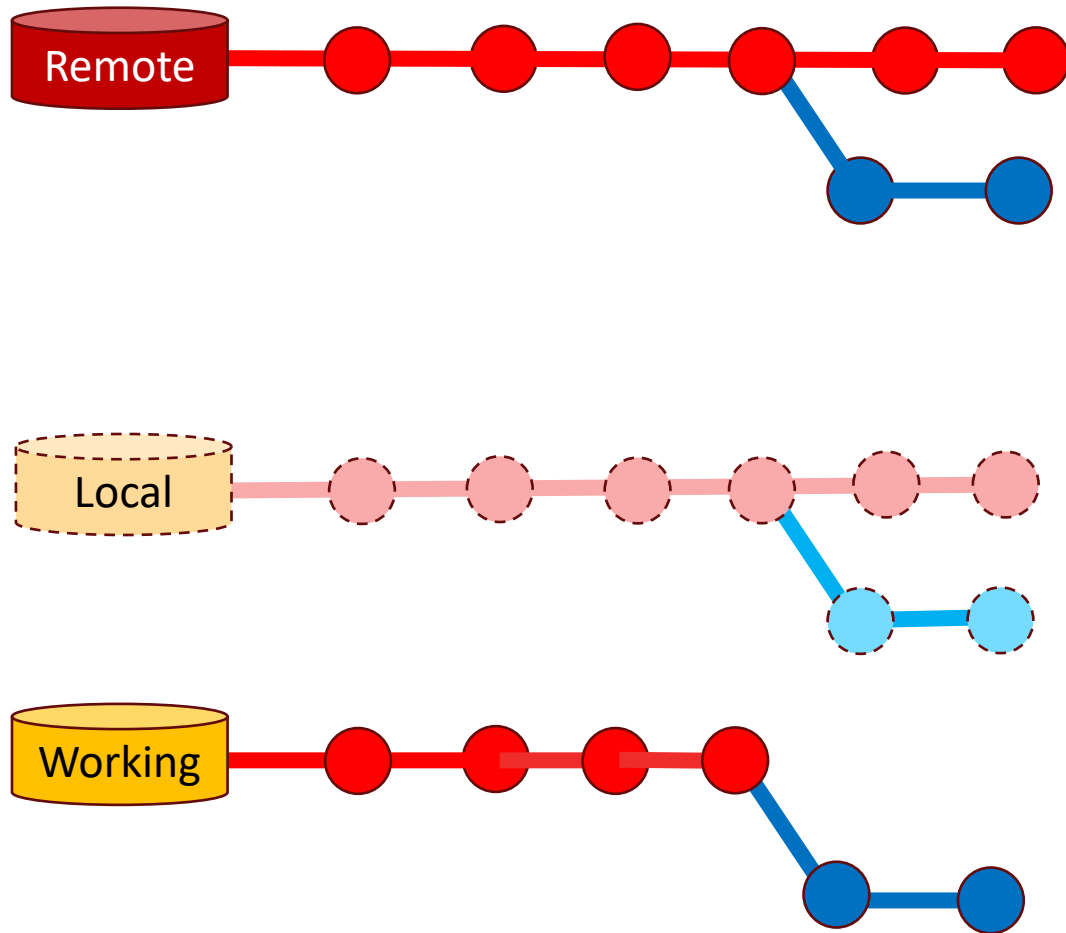
Workflow: Merge -2-



Which updates the "local" as well as the working directories

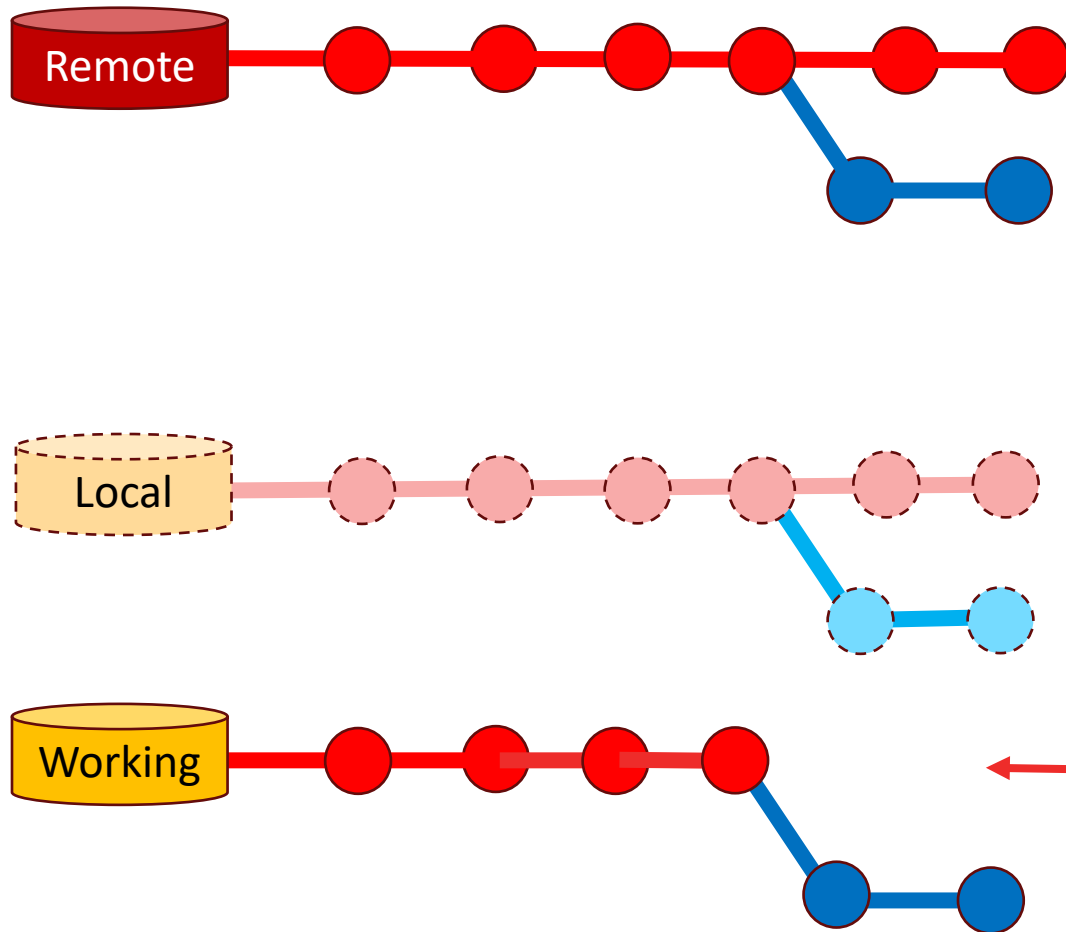
Using "pull" to update project (Project Update in IntelliJ)

Workflow: Merge -2-



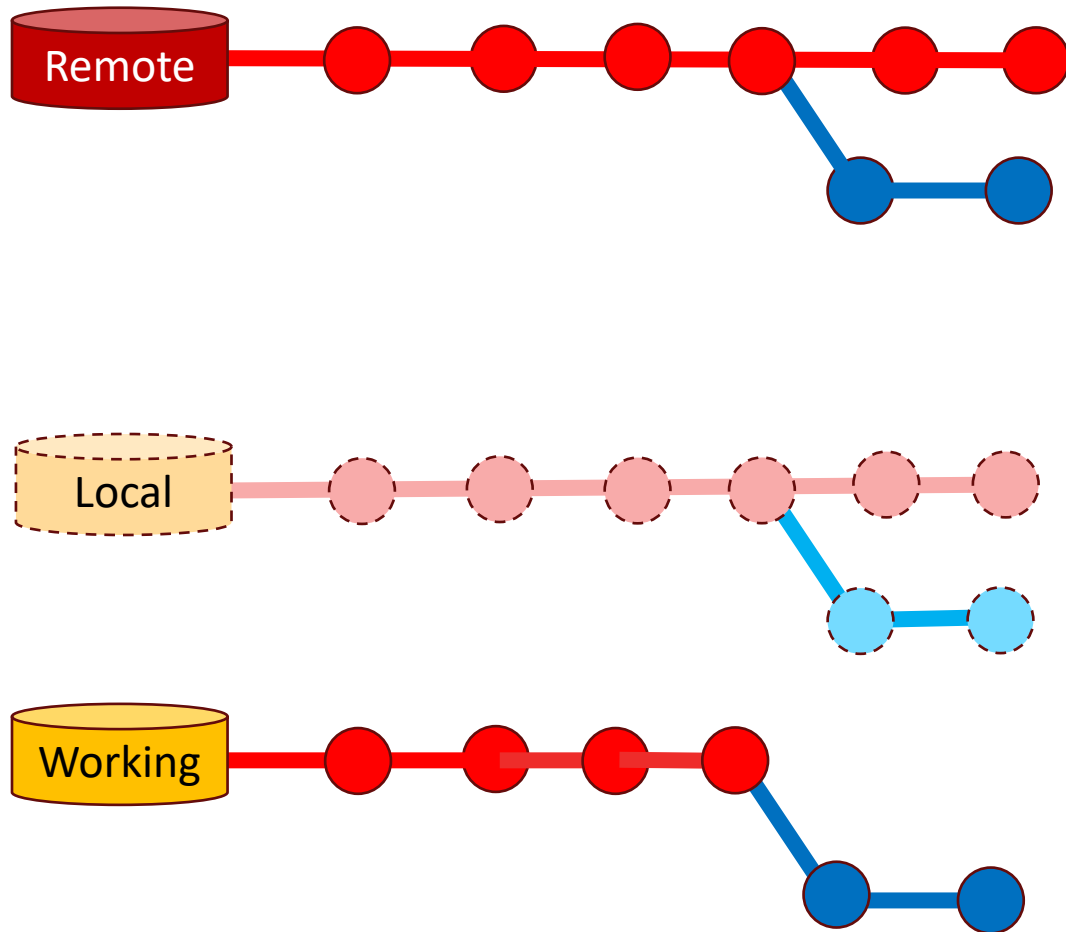
Barry then does a "checkout" to his own branch

Workflow: Merge -2-



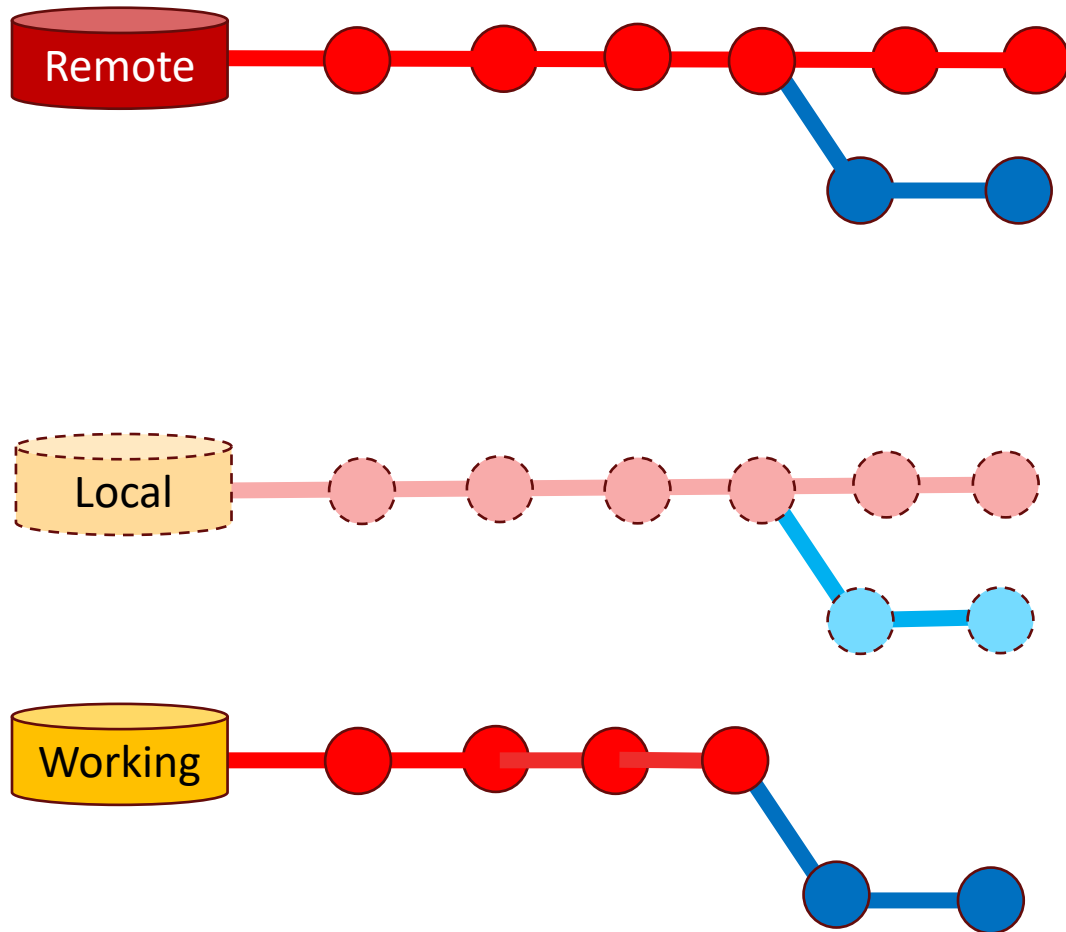
Where did the changes he pulled from remote go?

Workflow: Merge -2-



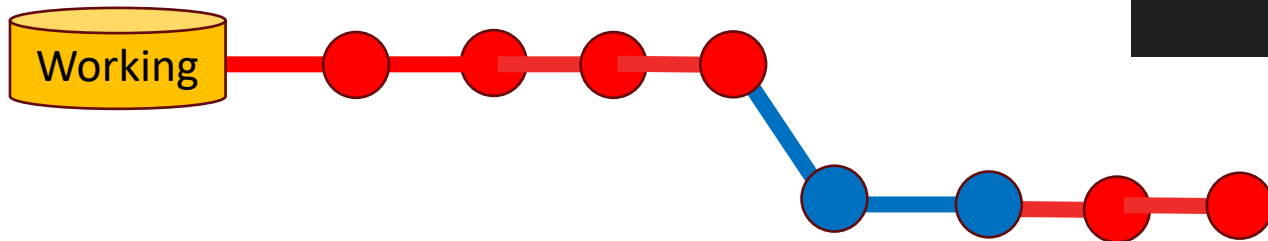
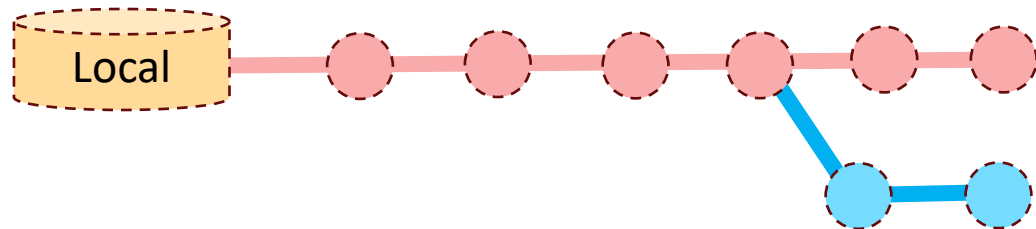
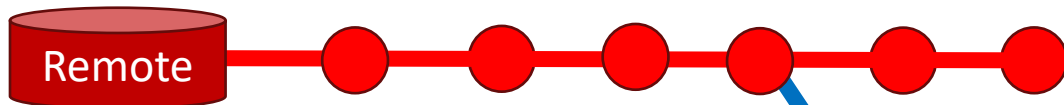
They are still here, working directory only shows the currently active branch.

Workflow: Merge -3-



Now Barry can merge the changes from the main branch to his branch

Workflow: Merge -3-

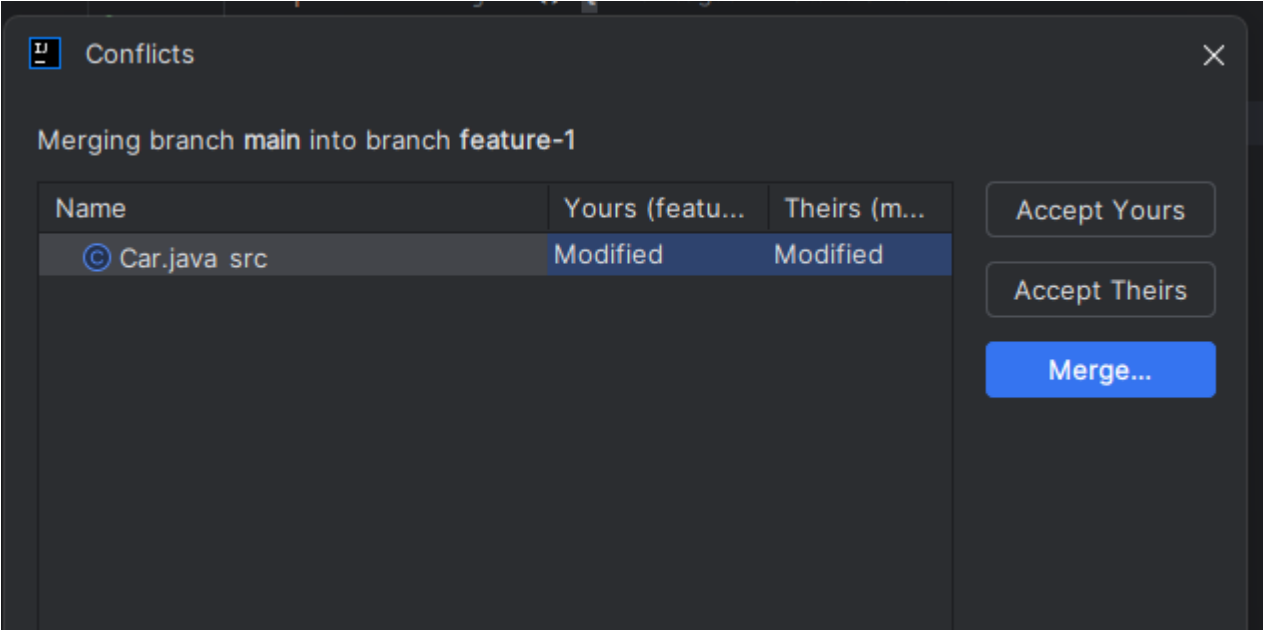
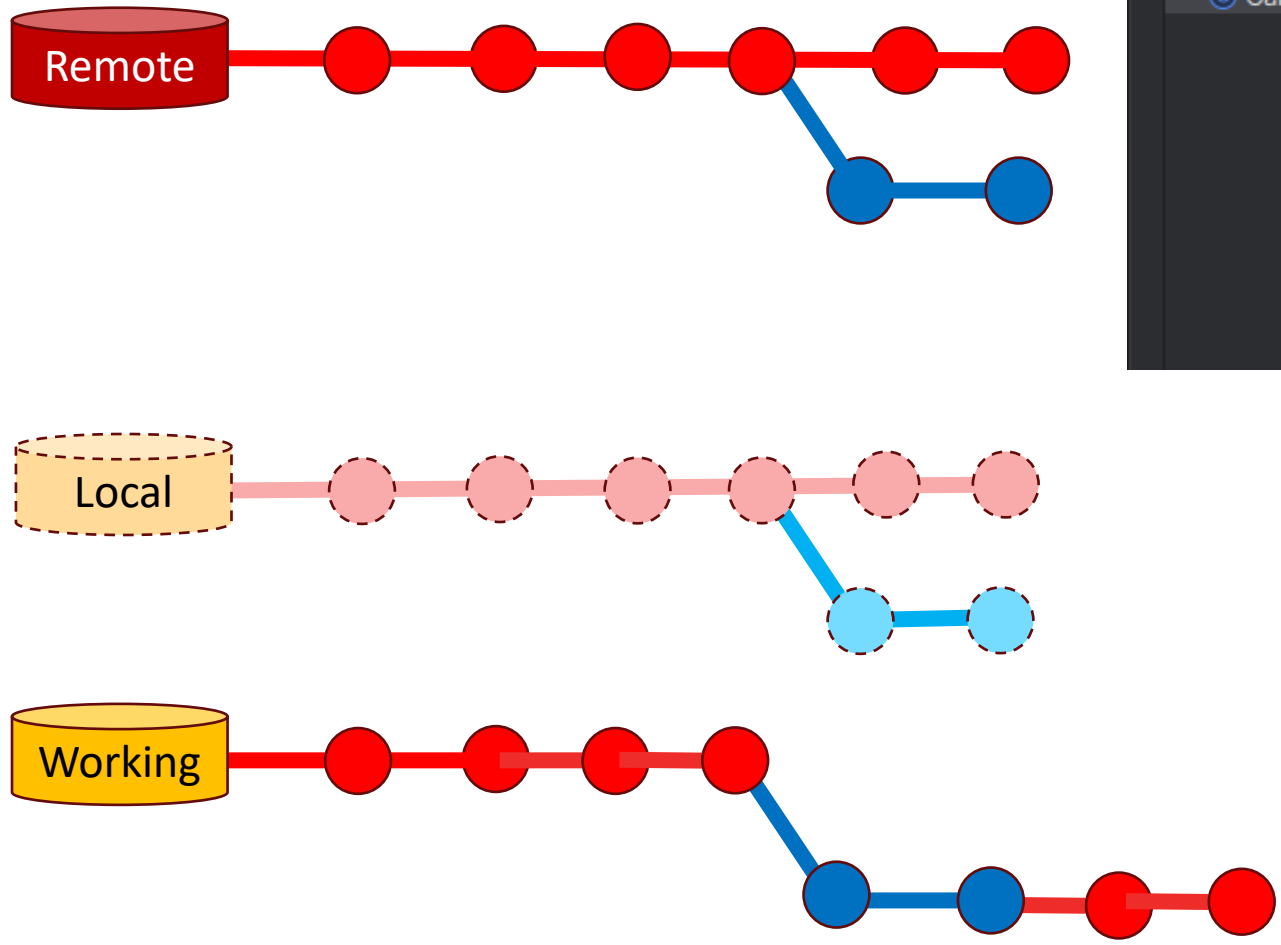


The screenshot shows an IDE interface with a menu open for the 'feature-1' branch. The menu options include 'Update Project...', 'Commit...', 'Push...', 'New Branch...', and 'Checkout Tag or Revision...'. The 'Recent' section shows 'feature-1' and 'main' under 'origin/'. The 'Local' section shows 'feature-1' and 'main'. A context menu is open over the 'main' branch in the 'Recent' section, with options like 'Checkout', 'New Branch from 'main'...', 'Checkout and Rebase onto 'feature-1'', 'Compare with 'feature-1'', 'Show Diff with Working Tree', 'Rebase 'feature-1' onto 'main'', 'Merge 'main' into 'feature-1'', 'Update', and 'Push...'. The 'Merge 'main' into 'feature-1'' option is highlighted.

Now Barry can merge the changes from the main branch to his branch

Merge Conflicts

Workflow: Merge -3-



Resolve conflicts if there are any.

<pre>public class Car extends Vehicle { private int x; private int y; public Car(int x, int y) { this.x = x; this.y = y; } public int getX() { return x; } public void setX(int x) { this.x = x; } public int getY() { return y; } public void setY(int y) { this.y = y; } }</pre>	1 2 3 4 5 6 7 8 9	1 2 3 4 5 6 7 8 9	<pre>public class Car extends Vehicle { private int x; private int y; public Car(int x, int y) { this.x = x; this.y = y; } }</pre>	1 2 3 4 5 6 7 8 9	<pre>public class Car extends Vehicle { private int x; private int y; public Car(int x, int y) { this.x = x; this.y = y; } public int getX() { if (x > y) { return x; } else { return y; } } }</pre>	1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18
---	---	---	---	---	---	---

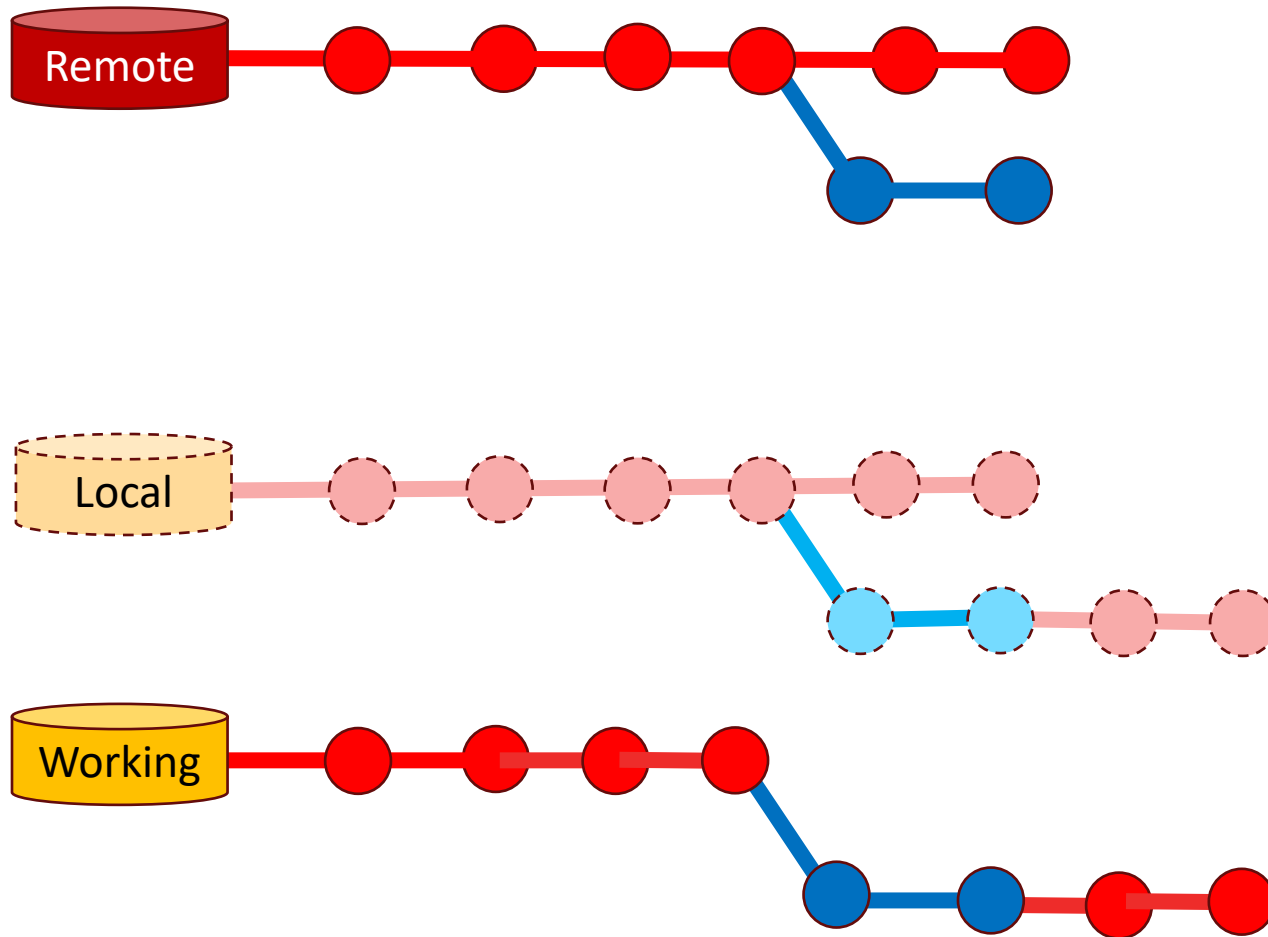
Accept
Ctrl+click to resolve conflict

You can click the double arrows to use your code or the code on main.

```
g\Java\CPSC219\F24\cpssc233ex1\src\Car.java
>> Left >>> All << Right ✎ Do not ignore Highlight words ✕ ↕ ⚙
Show Details Result Changes
e 1 1 public class Car extends Vehicle { 1
2 2 private int x; 2
3 3 private int y; 3
4 4
5 5 public Car(int x, int y) { 5
6 6 this.x = x; 6
7 7 this.y = y; 7
8 8 } 8
9 9 public int getX() { 9
10 10 // can't depend on y 10
11 11 return x; 11
12 12 } 12
13 13
14 14 public void setX(int x) { 15
15 15 this.x = x; 16
16 16 } 17
17 17
18 18 public int setY(int y) { 18
19 19 this.y = y; 19
20 20 } 20
21 21
22 22 public void setY(int y) { 23
23 23 this.y = y; 24
24 24 } 25
25 25
26 26 } 26
27 27 }
```

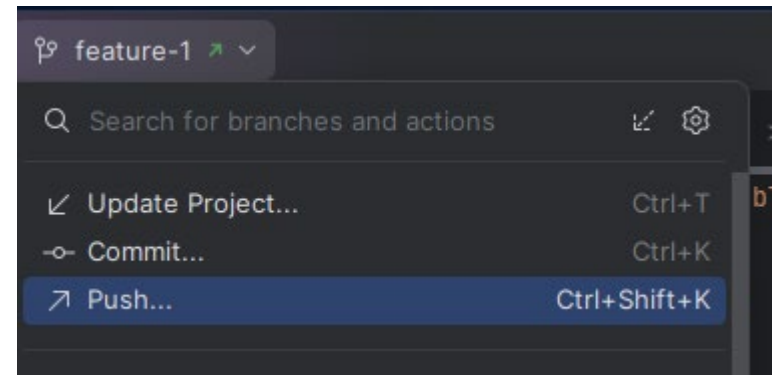
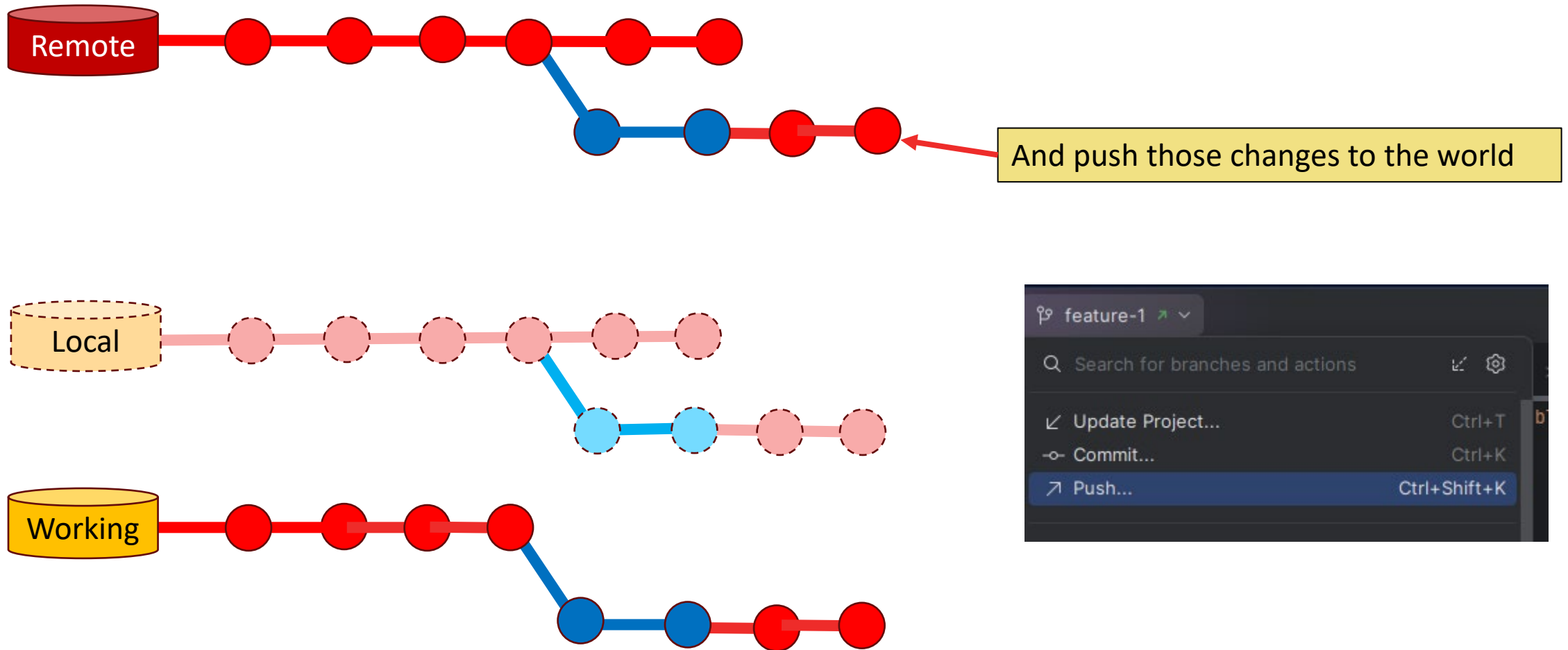
You can edit the text in the center.

Workflow: Merge -3-



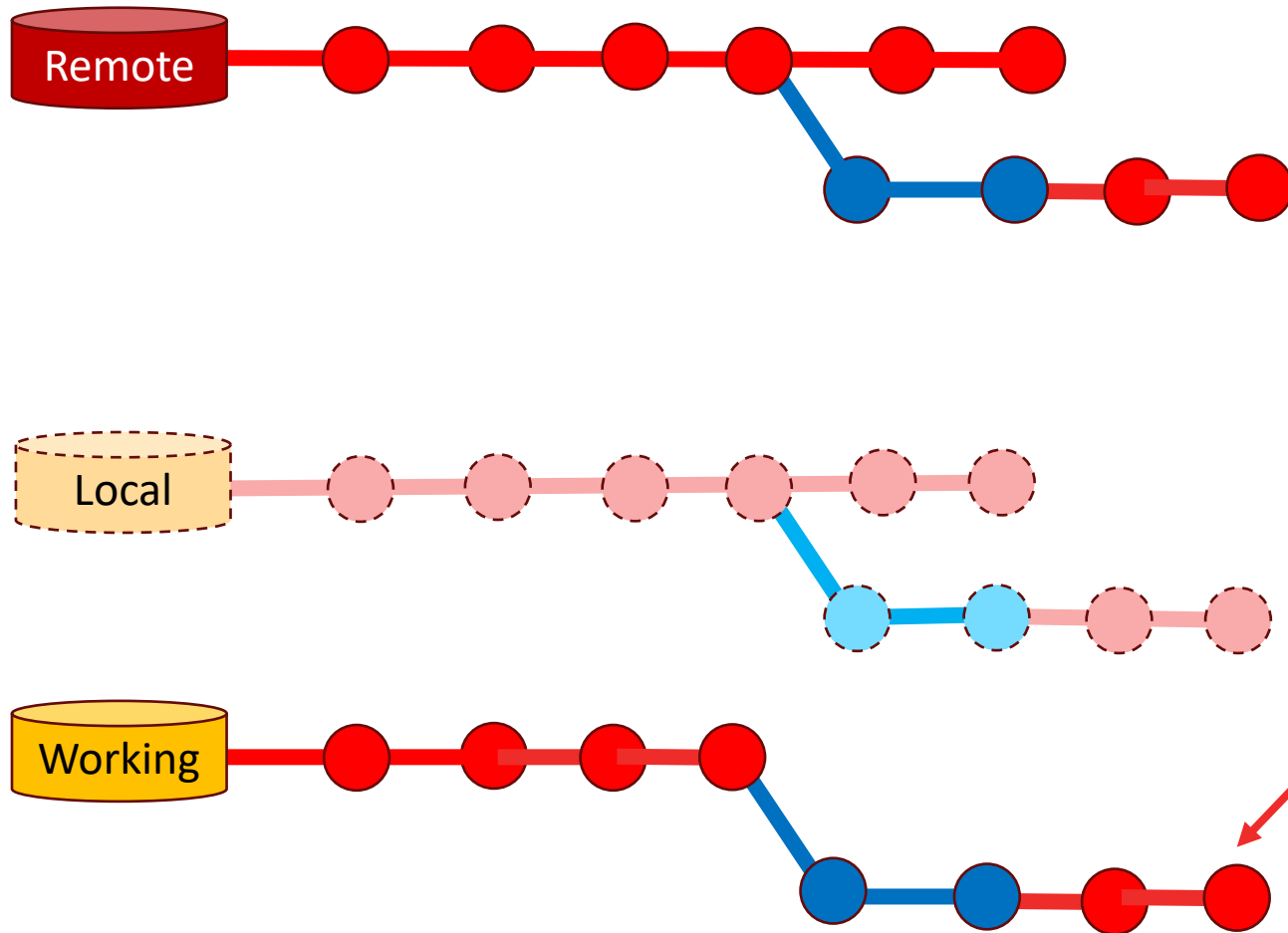
Commit changes (intellij does this automatically for you)

Workflow: Merge -3-



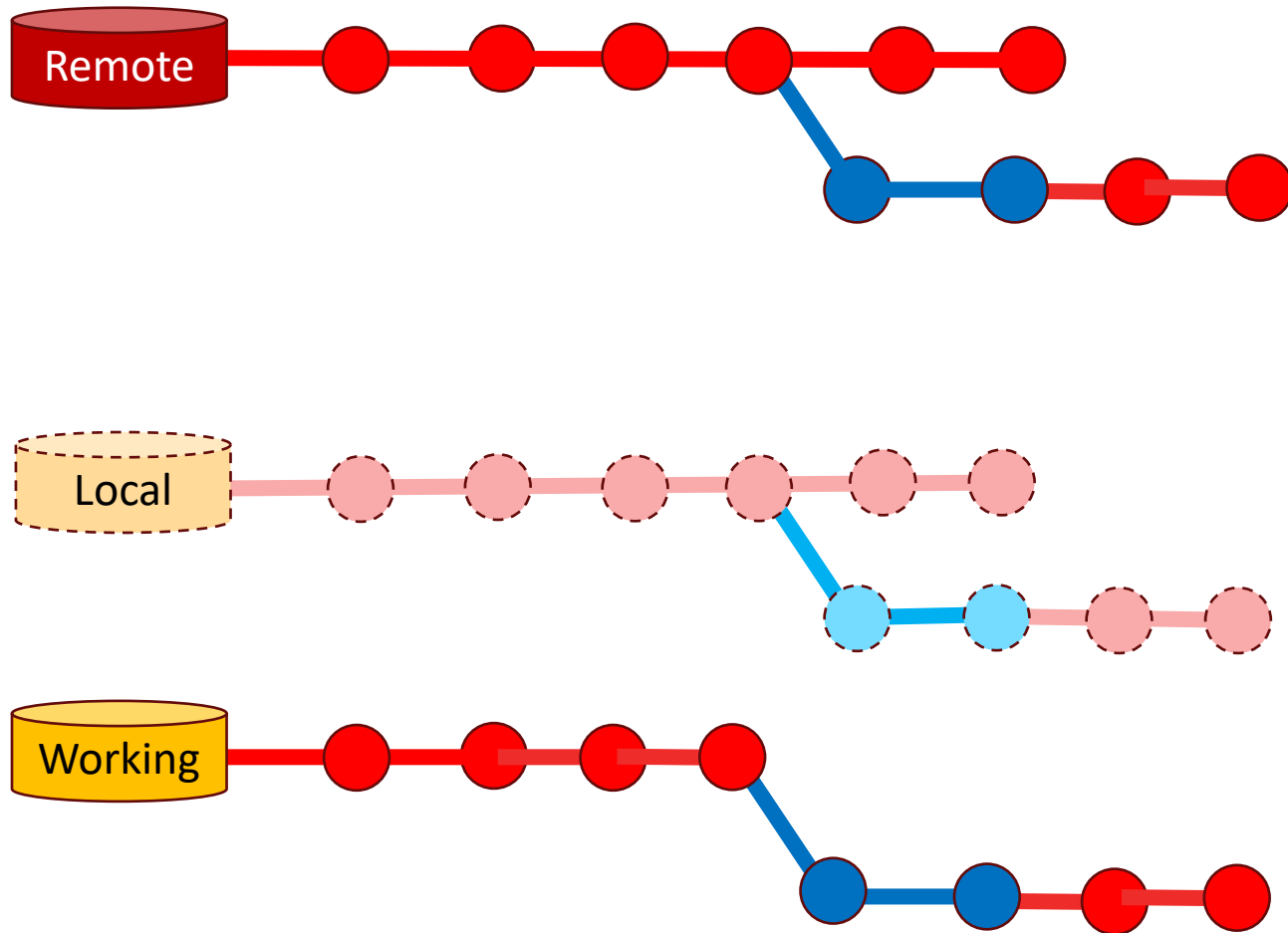
Pull requests

Workflow: Merge -4-

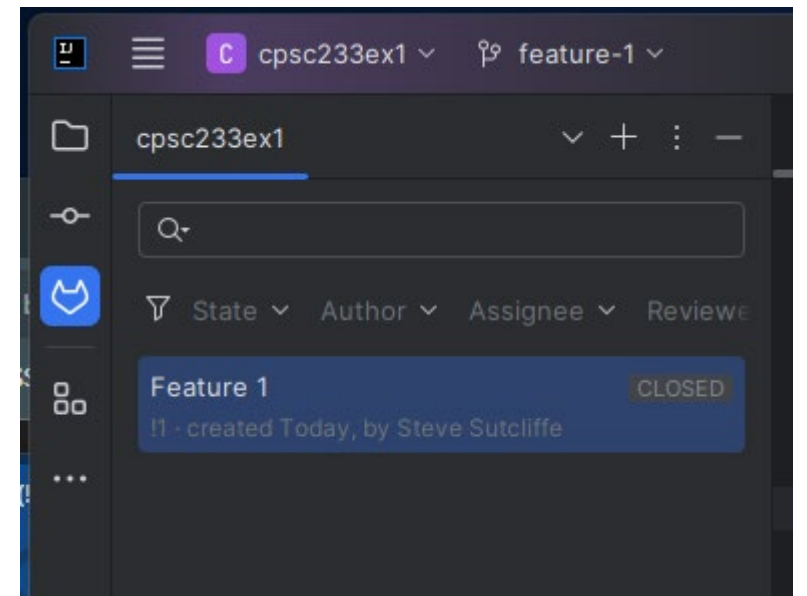
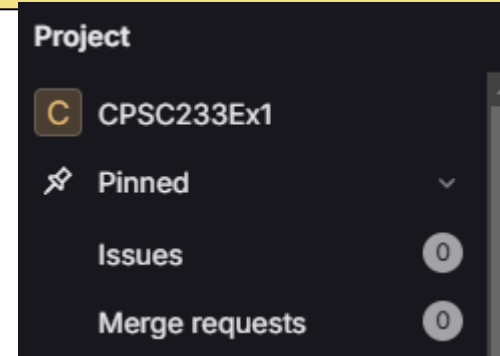


Once Barry is happy with the result, and wants to make his changes permanent

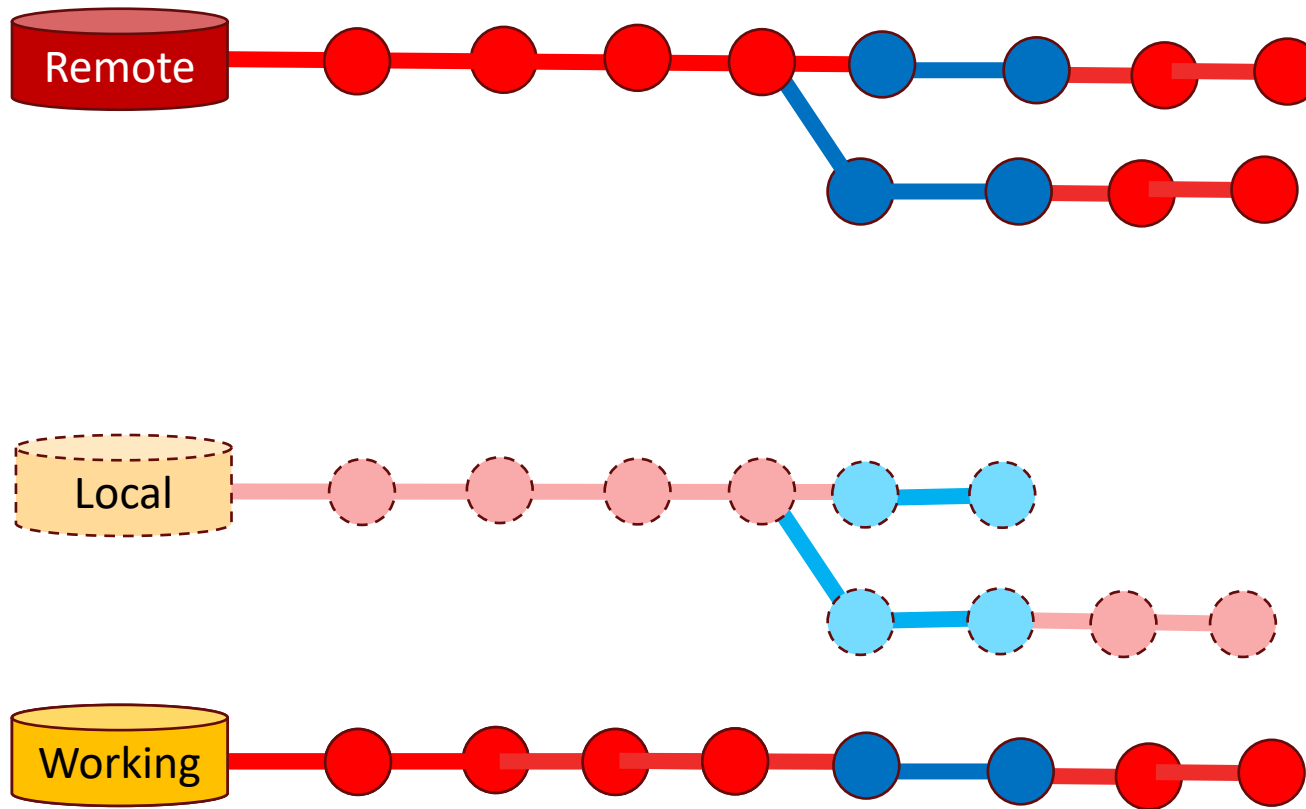
Workflow: Merge -4-



Barry will make a Pull Request (PR), called a Merge Request in IntelliJ, using the GitLab “Merge Requests” tab, or doing a Merge Request in IntelliJ



Workflow: Merge -4-



Then one member of Barry's team will Approve the PR to make the change permanent in remote

feature-1
Merged cpsc233Ex1Token requested to merge feature-1 into

Overview 0 Commits 3 Pipelines 0 Changes 1

0 0

8 Approval is optional

Merged by Steve Sutcliffe just now Revert

Merge details

- Changes merged into main with [d367237f](#) (commits were squashed).
- Did not delete the source branch.

Onward to ... System.

Jonathan Hudson
jwhudson@ucalgary.ca
<https://pages.cpsc.ucalgary.ca/~jwhudson/>

