Notes:

* In Git, ***commit*** is (almost) synonymous with project ***version***
* Commit saves changes to files – not the files themselves
  + great for text file – bad for other types of files
* If your RStudio Project is also on OneDrive/Google Drive
  + Turn off autosave (warning that always appears)
  + Avoid constant writing to a large data file
  + Add ***~\**** to ***.gitignore*** in the main project folder (so temp files are not sent in GitHub)

**File conflicts:**

File conflicts always happen when the same version of a file is edited in two different places and at least one person has already ***Pushed*** the edited file to GitHub.

Let's assume Ann is working on commit #17 and edited a file called ***stuff.R***. She then creates a new commit (#18) and ***Push***es the change to GitHub.

Bob, who is still working on commit #17, has also edited ***stuff.R***. Here are the scenarios:

1. Bob tries to ***Pull*** the latest commit from GitHub without creating a new commit
   * Bob will get a warning saying that the ***Pull*** cannot be completed until he either undoes or commits the change to ***stuff.R*** .
   * Bob can ***undo*** the changes by either:
     1. saving the file under a different name and deleting it
     2. Revert the file (right-click, ***Revert*** in the ***Git*** tab)
   * Bob can ***commit*** the changed file and then he can ***Pull*** from GitHub (scenario 3)
2. Bob creates a new commit (#18b) and tries to ***Push*** the changes to GitHub
   * Bob will get a warning that says that he does not have the most recent commit. He will need to ***Pull*** the changes first (scenario 3)
3. Bob creates a new commit (#18b) and tries to ***Pull*** the changes from GitHub
   * If the ***stuff.R*** files are mergeable, then Git will say that the two versions of the file have been merged into one and Bob can now ***Commit*** and ***Push*** the merged commit.
   * If the ***stuff.R*** files are not mergeable, then Bob will have to manually edit ***stuff.R*** and then send a merge instruction to Git. Once done, he can then ***Commit*** and ***Push*** the merged commit.

***Mergeable files*** means that the changes to the different ***stuff.R*** files are in different locations and Git can intelligently merge the changes into one file.

A conflict in a file will be shown like this:

<<<<<<< HEAD

# this is the code for the first file

=======

# this is the code for the second file

>>>>>>> master

**From Last Week**

**Recovering an old file commit**

Go to Git tab 🡪 History (the clock icon):

* top box shows all your project commits
  + the SHA is a unique identifier for the project commit
  + for this example, assume SHA is 9a45f121
* bottom box shows:
  + files changed in this commit
  + changes to the files (green lines: stuff added, red lines: stuff deleted)

To recover file:

* Go to the file changes section to the file you want to recover
* Click ***View file @ 9a45f121*** (top-right corner of file changes)
* File is opened in editor in new windows: click ***Save As***

The changed file is now in your current Project and reflected in the Git tab and can be saved to the next Commit.

**Recovering an old Project commit** (the safest way – assume SHA is 9a45f121)

In ***Terminal*** Tab type:

**git revert --no-commit 9a45f121..HEAD**

This command will revert all the files back to the 9a45f121 commit of your project

* ***--no-commit*** means a new version of your project is not automatically created
* **9a45f121..HEAD** means reverse all changes between 9a45f121 and HEAD, which is always the last commit

This command reverse edits the files that changed between the 9a45f121 commit and the current commit (HEAD). After you execute the command, all the files that changed will be put in the ***Git*** tab – just like when you edit files. You can now create a new commit with the reverted files.

If there are files ***you do not want to revert***, you can right-click on the file and choose ***Revert.*** This will revert the revert – and put the file back to the original state.