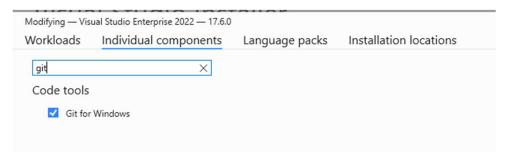
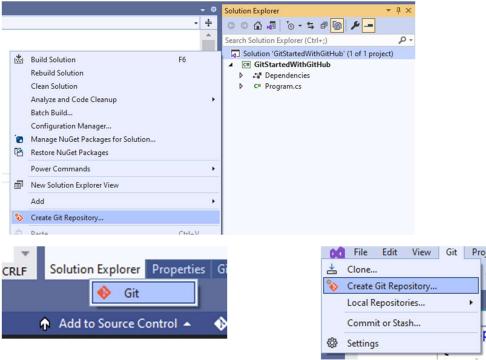
Getting Started with GitHub

Episode 1: Using Source Control

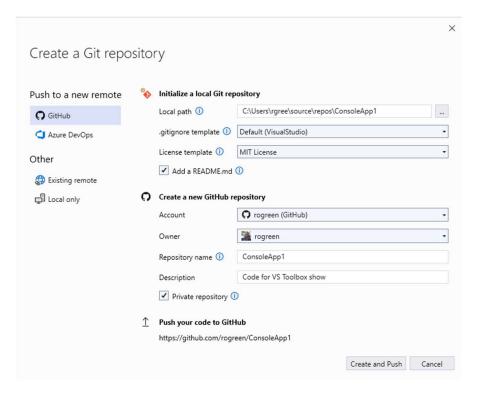
- 1. In Visual Studio Installer, click Modify.
- 2. On Individual Component tab, search for Git.



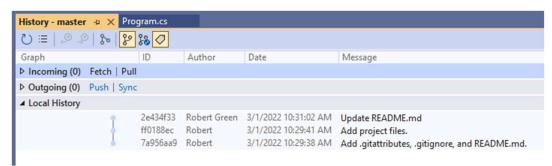
- 3. In Visual Studio, create Console app.
- 4. Review why you want to use source control:
 - a. Backup of code
 - b. Work on multiple machines
 - c. Support multiple devs working on code
 - d. Share code with others
- 5. Right-click solution and select **Create Git Repository** or select **Add to Source Control | Git** or select **Git | Create Git Repository**.



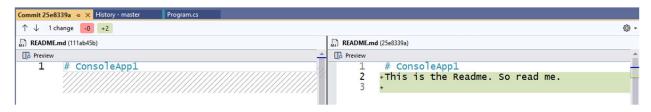
6. Fill out Create a Git Repository dialog.



- 7. Click Create and Push.
- 8. In GitHub, show the repo now exists.
- 9. Open .gitignore, license.txt and readme.md.
- 10. Update the Readme and commit.
- 11. In VS, select **Git | Open in File Explorer**.
- 12. Show .git folder, .gitattributes, .gitignore, readme.md.
- 13. Notice the readme is the original one.
- 14. In VS, select Git | Pull.
- 15. Notice the readme is now the current one.
- 16. Select Git | View Branch History.



17. Double click last commit to see what changes were made.



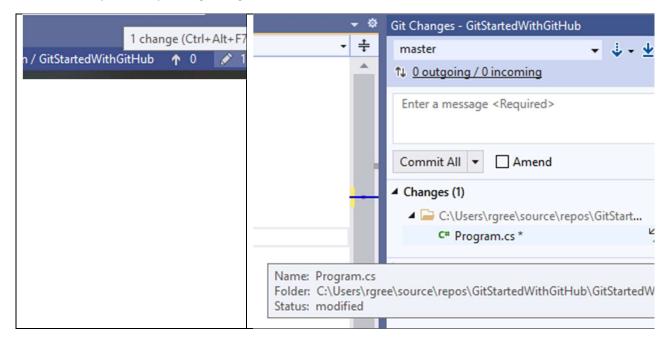
- 18. Close the solution.
- 19. Select **Git | Clone**.
- 20. In GitHub, go to main repo page.
- 21. Click **Code** and copy the URL.
- 22. Paste it back in Visual Studio.
- 23. Click Clone.
- 24. Select **Git | Open in File Explorer**.

Episode 2: Committing Code Changes

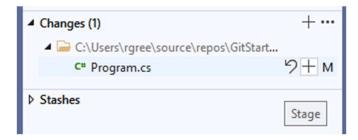
1. Make the following changes in Program.cs.

Console.WriteLine("Hello, Visual Studio Toolbox!");

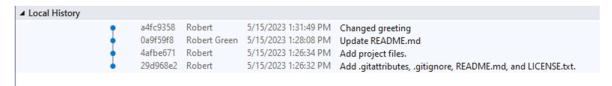
2. See that you have pending changes.



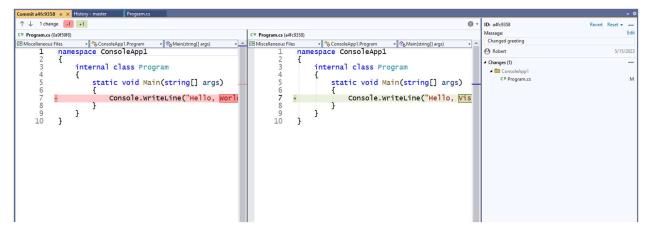
3. See that you can Undo changes and Stage/Stash changes.



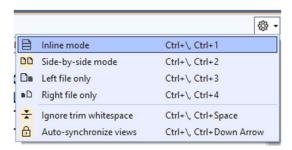
- 4. Enter a commit message and click **Commit All and Push**.
- 5. Show the changes are now in GitHub.
- 6. Click the commit message to see the commit.
- 7. In VS, view the branch history and see the commit.



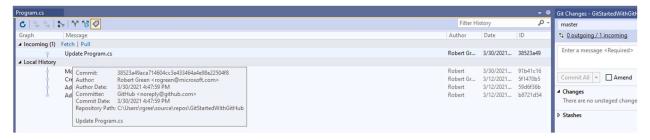
8. Double click the commit to see before and after. Notice that you can Revert.



9. You can switch from side by side to inline mode if you prefer.



- 10. In GitHub change the greeting in Program.cs and Commit. Enter text into Extended Description.
- 11. Switch to VS.
- 12. Fetch. Master does not show the change.
- 13. Click on 0 outgoing / 1 incoming and expand Incoming.



14. Double click Update Program.cs to compare the two versions.

```
Commit a4fc9358 Commit 67ddcd23 → X Git Repository - ConsoleApp1
↑ ↓ 2 changes -1 +2
C# Program.cs (a4fc9358) → C# Program.cs (67ddcd23)
C# Miscellaneous Files

→ ConsoleApp1.Program

    ◆ Main(string[] args)

            namespace ConsoleApp1
      3
                 internal class Program
      4
                       static void Main(string[] args)
      5
      6
                            Console.WriteLine("Hello, Visual Studio Toolbox!");
                            Console.WriteLine("Hello, Visual Studio developers!");
      8
                 }
      9
     10
     11
```

15. Click Pull.



16. Refresh the branch history to see both changes to Program.cs (one in VS and one in GitHub).

Local History					
	•	73fbe2ed	Robert Green	5/4/2023 8:12:45 PM	Update Program.cs
	•	85a7bd	Robert	5/4/2023 8:10:13 PM	Changed startup greeting
	•	41e17cd5	Robert Green	5/4/2023 8:04:12 PM	Update README.md
	•	7b096a49	Robert	5/4/2023 8:03:14 PM	Add project files.
	•	05ce6b34	Robert		Add .gitattributes, .gitignore, README.md, and LICENSE.tx

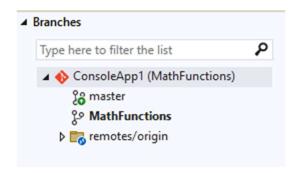
Episode 3: Working in branches

Create work in branches

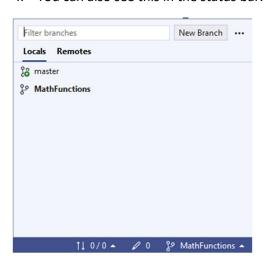
- Create new branches to isolate changes for a feature or a bug fix from your master branch and other work.
- Git does not create multiple copies of your source when working with branches—it uses the history information stored in commits to recreate the files on a branch when you start working on it.
- Isolating work in branches makes it very simple to change what you are working on by simply changing your current branch.



- 1. Select **Git | New Branch** or select **Git | Manage Branches** and then right-click main on master and select **New Local Branch From**
- 2. Enter MathFunctions as the branch name and Create.
- 3. Select **Git | Manage Branches**. See that MathFunctions is the current branch.



4. You can also see this in the status bar.

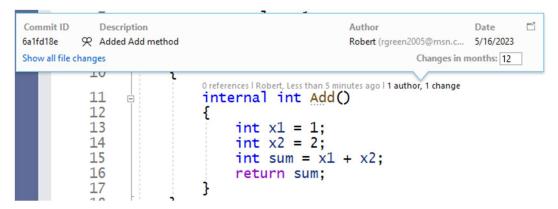


- 5. Add Math class.
- 6. Add the following method

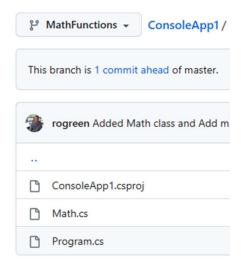
```
internal int Add()
{
    int x1 = 1;
    int x2 = 2;
    sum = x1 + x2;
```

```
return sum;
}
```

- 7. Commit and push. Notice that you pushed the branch.
- 8. You can see what happened in the CodeTips.



- 9. In GitHub, see that the master branch does not have your new code.
- 10. Switch to the MathFunctions branch and see your code.

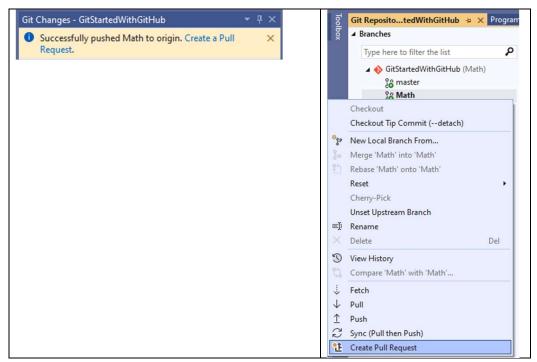


- 11. In VS, switch to Master branch and see that the changes disappear from VS.
- 12. Switch back to MathFunctions.

Create a pull request

Create a pull request The pull request is the collaborative process that lets the rest of the team discuss changes in a branch and agree to merge them once everyone approves. Use pull requests to get early feedback from others on work in progress, even if you're not ready to merge the changes into another branch. You must resolve merge conflicts before you can commit a pull request.

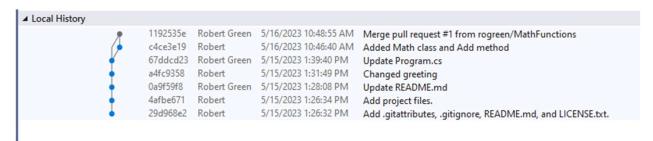
1. In VS, click Create a Pull Request



2. In GitHub you can request a review from or assign this to someone



- 3. Click Create pull request.
- 4. Click Merge pull request and then Confirm merge. Notice you can delete the branch.
- 5. Show that master branch now has the changes.
- 6. In VS, change the branch to master. Notice changes aren't there.
- 7. Select **Git | Pull**. Your local copy is now in sync.



Episode 4: Resolving merge conflicts

Resolve merge conflicts

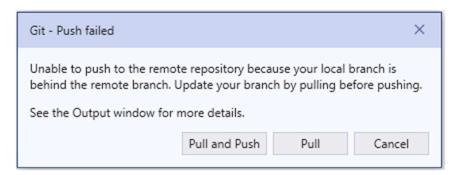
- When you merge one branch into another, file changes from commits in one branch can conflict with the changes the other.
- Git attempts to resolve these changes by using the history in your repo to determine what the merged files should look like.
- When it isn't clear how to merge changes, it halts the merge and tells you which files conflict.



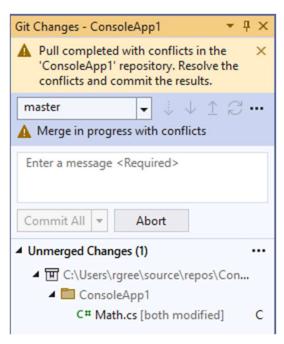
- 1. In GitHub, make sure master is the current branch.
- 2. Change int to var. Commit
- 3. In VS, make the following change:

internal double Add(double x1, double x2) => x1 + x2;

- 4. Commit and push
- 5. Push fails.



6. Click Pull.

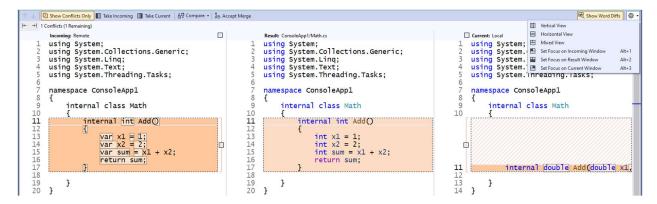


```
1 File contains merge conflicts. Open Merge Editor
C# ConsoleApp1

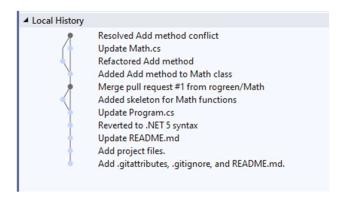
→ ConsoleApp1.Math

        1
             □using System;
        2
              using System.Collections.Generic;
        3
              using System.Linq;
        4
              using System.Text;
        5
              using System. Threading. Tasks;
        6
        7
             □namespace ConsoleApp1
        8
              {
                    0 references | Robert, 13 minutes ago | 1 author, 3 changes
        9
                    internal class Math
       10
       11
              Oreferences | Robert, Less than 5 minutes ago | 1 author, 1 change internal double Add (double x1, double x2) => x1 + x2;
       12
       13
       14
                         internal int Add()
       15
       16
                              var x1 = 1;
       17
                              var x2 = 2;
       18
                              var sum = x1 + x2;
       19
                              return sum;
       20
       21
                         f904f032107d696f0f5110bcdbb8b6831fc71273
       22
       23
                    }
       24
```

- 7. Click Open Merge Editor.
- 8. Compare Files

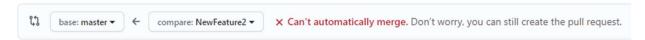


- 9. Click Take Current or use check boxes.
- 10. Click Accept Merge.
- 11. Commit and Push.
- 12. Confirm that GitHub now has refactored Add method.

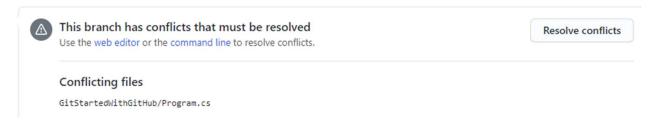


Episode 5: More on merge conflicts

- 1. Select Git | New Branch.
- 2. Enter **Utilities** as the branch name and **Create**.
- 3. Add a Utilities class.
- 4. Add code to call Utilities in Main.
- 5. Commit and push.
- 6. In GitHub add a Services class and add code to call Services in Main. Commit.
- 7. Create a pull request in VS.
- 8. Notice that you are warned there are conflicts



- 9. Click Create pull request.
- 10. You are alerted to conflicts.



11. Click **Resolve conflicts** to see the problem.

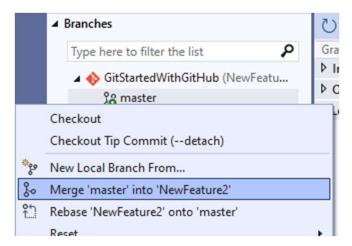
```
// Add code here to call NewFeature2

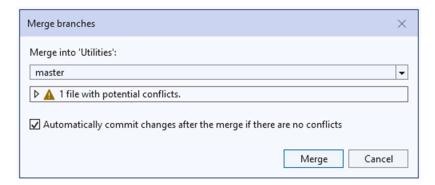
// Add code here to call NewFeature4

------
// Add different code here to call NewFeature2

>>>>> master
}
```

- 12. In VS, make master the current branch.
- 13. Pull
- 14. Make Utilities the current branch.
- 15. In Branches dialog, right click master and select Merge master into Utilities.





16. Click Merge.



- 17. Click Resolve the conflicts.
- 18. In the Git Changes window, double click **Program.cs** to see the conflict.
- 19. Click Open Merge Editor.
- 20. Select both lines of code. Note that the order you click controls what order they are added.
- 21. Click Accept Merge.
- 22. Commit Staged and Push.
- 23. Go to Pull Requests in GitHub and click Merge pull request.
- 24. Click Confirm merge.
- 25. Back in VS change branch to master and pull.
- 26. Confirm Program in master calls both Utilities and Services.

