Getting Started with GitHub

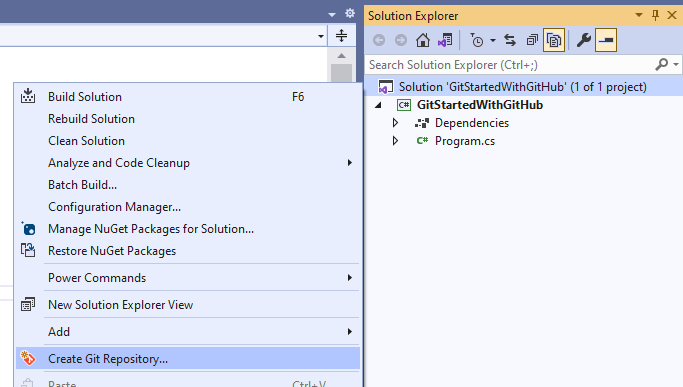
# Episode 1: Using Source Control

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

A screenshot of a computer

Description automatically generated with medium confidence

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



|  |  |
| --- | --- |
| Graphical user interface, text, application, email  Description automatically generated |  |

1. Fill out Create a Git Repository dialog.

Graphical user interface, text, application, email

Description automatically generated

1. Click **Create and Push**.
2. In GitHub, show the repo now exists.
3. Open .gitignore, license.txt and readme.md.
4. Update the Readme and commit.
5. In VS, select **Git | Open in File Explorer**.
6. Show .git folder, .gitattributes, .gitignore, readme.md.
7. Notice the readme is the original one.
8. In VS, select **Git | Pull**.
9. Notice the readme is now the current one.
10. Select **Git | View Branch History**.

Table

Description automatically generated with medium confidence

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

A screenshot of a computer

Description automatically generated with medium confidence

1. Close the solution.
2. Select **Git | Clone**.
3. In GitHub, go to main repo page.
4. Click **Code** and copy the URL.
5. Paste it back in Visual Studio.
6. Click **Clone**.
7. Select **Git | Open in File Explorer**.

**END**

# Episode 2: Committing Code Changes

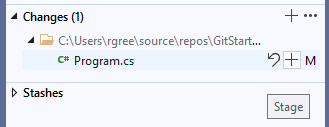
1. Make the following changes in Program.cs.

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

1. See that you have pending changes.

|  |  |
| --- | --- |
|  |  |

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



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

A screenshot of a computer

Description automatically generated with low confidence

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

A screen shot of a computer

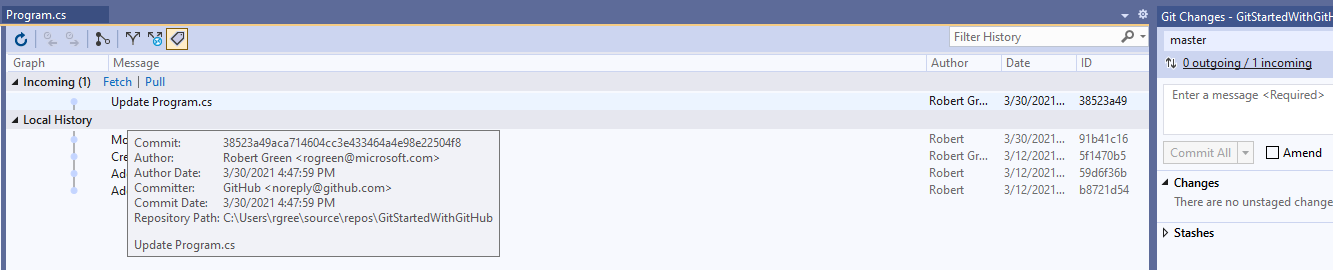
Description automatically generated with medium confidence

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

Graphical user interface, text, application

Description automatically generated

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



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

A computer screen shot of a program

Description automatically generated with low confidence

1. Click **Pull**.

Graphical user interface, text, application, table

Description automatically generated

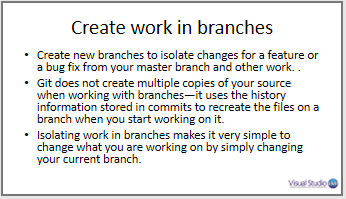
1. Refresh the branch history to see both changes to Program.cs (one in VS and one in GitHub).

Text

Description automatically generated

**END**

# Episode 3: Working in branches



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.

Graphical user interface, text, application, email

Description automatically generated

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

Graphical user interface, text, application, email

Description automatically generated

1. Add **Math** class.
2. Add the following method

internal int Add()

{

    int x1 = 1;

    int x2 = 2;

    sum = x1 + x2;

    return sum;

}

1. Commit and push. Notice that you pushed the branch.
2. You can see what happened in the CodeTips.

A screenshot of a computer

Description automatically generated with medium confidence

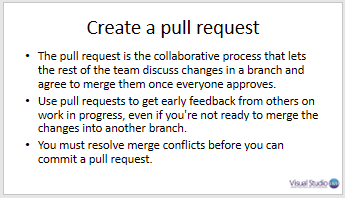
1. In GitHub, see that the master branch does not have your new code.
2. Switch to the MathFunctions branch and see your code.

Graphical user interface, text, application, email

Description automatically generated

1. In VS, switch to Master branch and see that the changes disappear from VS.
2. Switch back to MathFunctions.

# Create a pull request



1. In VS, click Create a Pull Request

|  |  |
| --- | --- |
|  |  |

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

|  |  |
| --- | --- |
|  |  |

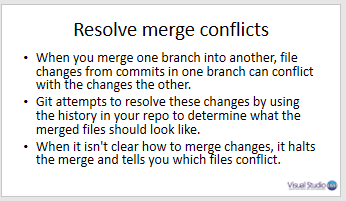
1. Click **Create pull request**.
2. Click **Merge pull request** and then **Confirm** **merge**. Notice you can delete the branch.
3. Show that master branch now has the changes.
4. In VS, change the branch to master. Notice changes aren’t there.
5. Select **Git | Pull**. Your local copy is now in sync.

A screenshot of a computer

Description automatically generated with medium confidence

**END**

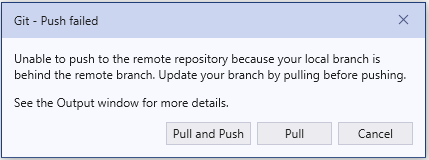
# Episode 4: Resolving merge conflicts



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;

1. Commit and push
2. Push fails.



1. Click **Pull**.

Graphical user interface, text, application, chat or text message

Description automatically generated

Graphical user interface, text, application

Description automatically generated

1. Click **Open Merge Editor**.
2. Compare Files

Calendar

Description automatically generated

1. Click **Take Current** or use check boxes.
2. Click **Accept Merge**.
3. Commit and Push.
4. Confirm that GitHub now has refactored Add method.

Graphical user interface, text

Description automatically generated

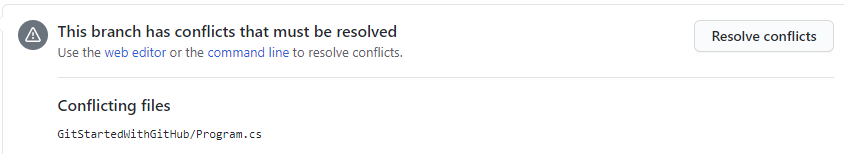
**END**

# Episode 5: More on merge conflicts

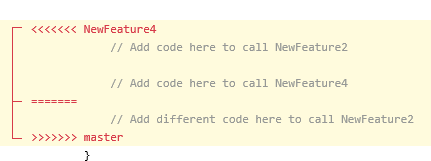
1. Select **Git | New Branch**.
2. Enter **Utilities** as the branch name and **Create**.
3. Add a **Utilities** class.
4. Add code to call Utilities in Main.
5. Commit and push.
6. In GitHub add a Services class and add code to call Services in Main. Commit.
7. Create a pull request in VS.
8. Notice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



1. Click **Create pull request**.
2. You are alerted to conflicts.



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



1. In VS, make master the current branch.
2. Pull
3. Make Utilities the current branch.
4. In Branches dialog, right click master and select **Merge master into Utilities**.

Graphical user interface, text, application, email

Description automatically generated

A screenshot of a computer error

Description automatically generated with low confidence

1. Click **Merge**.



1. Click **Resolve the conflicts**.
2. In the Git Changes window, double click **Program.cs** to see the conflict.
3. Click **Open Merge Editor**.
4. Select both lines of code. Note that the order you click controls what order they are added.
5. Click **Accept Merge**.
6. Commit Staged and Push.
7. Go to Pull Requests in GitHub and click **Merge pull request**.
8. Click **Confirm merge**.
9. Back in VS change branch to master and pull.
10. Confirm Program in master calls both Utilities and Services.

A screenshot of a computer

Description automatically generated with medium confidence

**END**