Getting Started with GitHub using VS 2019

This document is a copy from an Azure DevOps Lab.

You can, if wanted, use a virtual machine to do the exercise, but it is not needed. All the software is normally on your pc.

- I was not able to download the virtual machine, the moment I created this document.

This is the url: [Getting Started with GitHub using Visual Studio 2019]

I've changed some text and the layout to make it more clear.

You can choose to work with the online page, or with this document. The online page, assumes you have a virtual machine.

It is also possible that some screens look different in your version. I've put the screens of the latest version available for Visual Studio 2019 at January 2022.

We are using some code from GitHub. The goal is not to understand the code. The goal is explaining GitHub functionality also within Visual Studio 2019.

Installation of GitHub

- You have GitHub in a browser. (Working in the cloud)
  - https://github.com/
- You have also GitHub on the desktop. (Working locally).
  - https://desktop.github.com/.

Create an account for GitHub

It all starts with having an account to work in the cloud.

- There is no reason not to have one.

Notes

Setting up GitHub in Visual Studio 2019

- Start “Visual Studio Installer”.
- Select the correct version if you have more than one.
- Hit the button “Modify”.
- You have another window.
- Select the tab “Individual components”.
- Mark “Git for Windows”.
- Mark “GitHub extension for Visual Studio”.
- Hit the button “Modify” to install.

Installation of Git, Hub or Git Bash or Git for Windows

- If you want (not used in the classroom).
- This is part of GitHub desktop.
  - It is working with Git in a command line in stead of a GUI.
- A nice cheatsheet to have the basic commands.
  - https://education.github.com/git-cheat-sheet-education.pdf
  - Git - Reference (git-scm.com)

Alternatives

- BitBucket.
- Visual Source Safe.
- …

Installation movie

There is a Youtube movie of about 20 minutes that explains it good. Just for information.

- https://www.youtube.com/watch?v=J_Clau1bYco

Notes

What is GitHub and Git?

It is a versioning tool and a collaboration tool for files and code.

It allows you to organize within your team.

- Versioning of code or files.
- Creating and maintaining branches.
- Creating and maintaining releases.
- It's a part of tooling to have DevOps.

Working with a versioning tool is necessary for all your projects. You never know what will happen with your work.

Working locally with versioning is not a good idea.

When your PC crashes, and your backup is on the same PC, you will have a problem.

- You are responsible of not losing months of work.

Adding an existing solution to GitHub

- Right-click the solution.
- Select "Create Git Repository …".
- Fill in the screens.

Disconnecting your project from GitHub

- Remove from your solution 2 files.
  - Remove ".gitattributes."
  - Remove ".gitignore."
- Remove from your solution the folder ".git".

Notes

How GitHub fits in

GitHub is a Git hosting repository that provides developers with tools to ship better code through command line features, issues (threaded discussions), pull requests, code review, or the use of a collection of free and for-purchase apps in the GitHub Marketplace.

With collaboration layers like the GitHub flow, a community of 15 million developers, and an ecosystem with hundreds of integrations, GitHub changes the way software is built.

How GitHub works

GitHub builds collaboration directly into the development process. Work is organized into repositories, where developers can outline requirements or direction and set expectations for team members.

Then, using the GitHub flow, developers simply create a branch to work on updates, commit changes to save them, open a pull request to propose and discuss changes, and merge pull requests once everyone is on the same page.

Notes

The GitHub flow

The GitHub flow is a lightweight, branch-based workflow built around core Git commands used by teams around the globe.

The GitHub flow has six steps, each with distinct benefits when implemented:

1. Create a branch: Topic branches created from the canonical deployment branch (usually master) allow teams to contribute to many parallel efforts. Short-lived topic branches, in particular, keep teams focused and results in quick ships.
2. Add commits: Snapshots of development efforts within a branch create safe, revertible points in the project's history.
3. Open a pull request: Pull requests publicize a project's ongoing efforts and set the tone for a transparent development process.
4. Discuss and review code: Teams participate in code reviews by commenting, testing, and reviewing open pull requests. Code review is at the core of an open and participatory culture.
5. Merge: Upon clicking merge, GitHub automatically performs the equivalent of a local 'git merge' operation. GitHub also keeps the entire branch development history on the merged pull request.
6. Deploy: Teams can choose the best release cycles or incorporate continuous integration tools and operate with the assurance that code on the deployment branch has gone through a robust workflow.

Notes

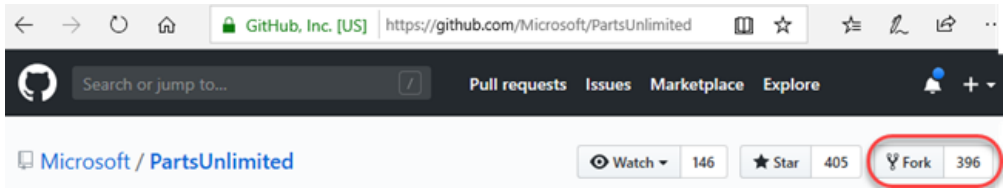


Figure 1: Fork in GitHub.

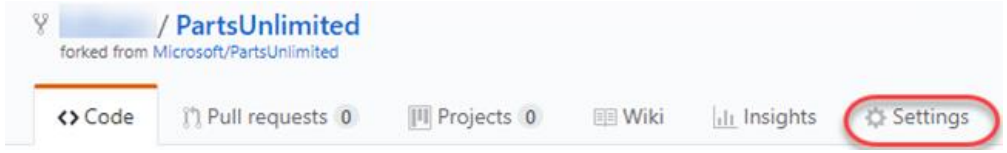


Figure 2: Settings in GitHub Repository.

Notes

.....

.....

.....

.....

.....

.....

.....

COPY PASTE)

Exercise 1: Getting Started with GitHub using Visual Studio 2019

The goal of this lab is that you follow this step by step as exercise. This must be done after the explanation of what we are doing in normal life 😊.

The explanation is a simple example. This step by step method is a more complex exercise.

Task 1: Setting up a GitHub project

- We will copy (fork) an existing project, so we can do all the functionalities as if you are working with a team who has created some software.
- We will do some changes (fixing an issue) in that project to explain how you can use versioning and doing a pull request.

1. Click on the link here below.
   https://github.com/Microsoft/PartsUnlimitedE2E
2. GitHub will be started in a browser.
3. Log in, with your account when you are not logged in yet (and if you have an account).
4. Fork the project into your own account. (*See Figure 1*)
5. Forking is making a copy of a project when you are not a contributor. Let's say, part of the programming team.
6. Wait a bit, until everything is loaded into your GitHub repository. You are creating a copy of the project, because you don't have the rights to change this Microsoft Project.
   The starting point is this copy. As if you were the creator of this software.
7. Select the "Settings" tab. (*See Figure 2*)

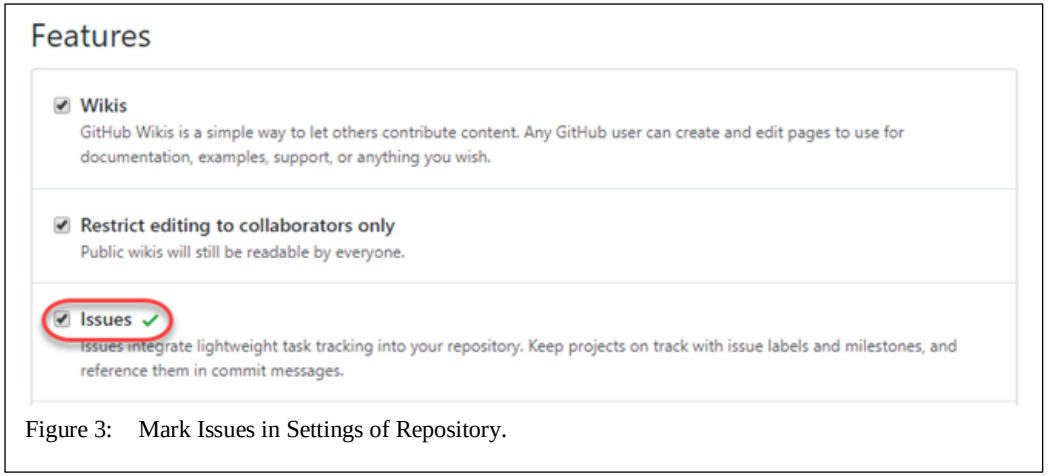


Figure 3: Mark Issues in Settings of Repository.

8. Enable “Issues” by checking its box. (*SEE FIGURE 3*)
9. Select the “Issues” tab. (*SEE FIGURE 4*)

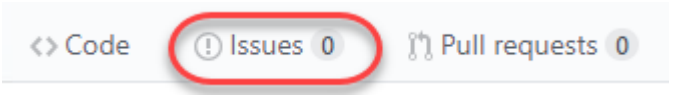


Figure 4: Select Issues in Repository.

10. Click “New issue”. (*SEE FIGURE 5*)

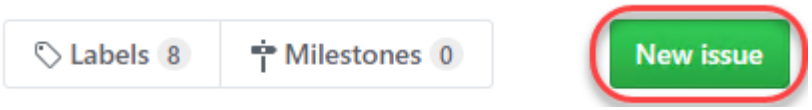


Figure 5: Select New issue

11. Create a new issue called “Update to v2.0” and click “Submit new issue”. (*SEE FIGURE 6*)

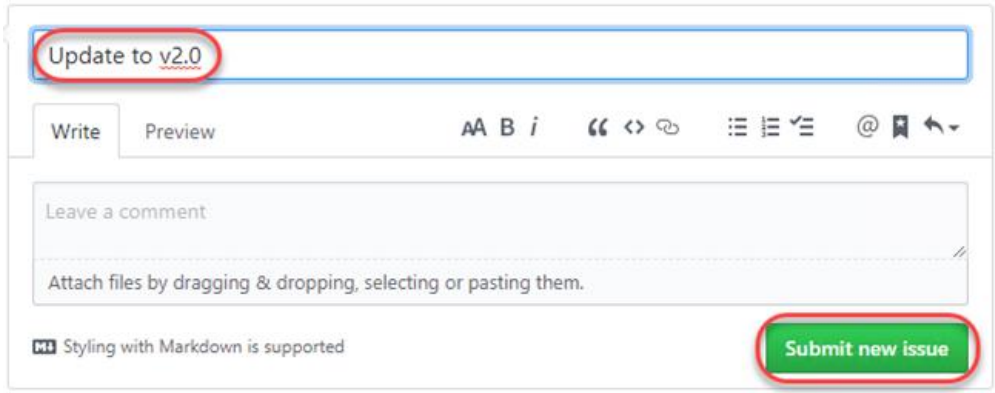


Figure 6: Type title of new issue and Submit new issue.

Notes

COPY PASTE)

Fixing this issue will be the focus of this lab.

12. Note the unique Id of the newly created issue. (*SEE FIGURE 7*)

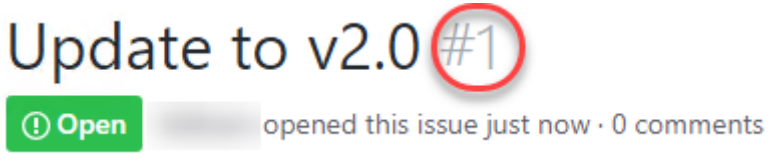


Figure 7: The unique identifier of the issue.

Notes

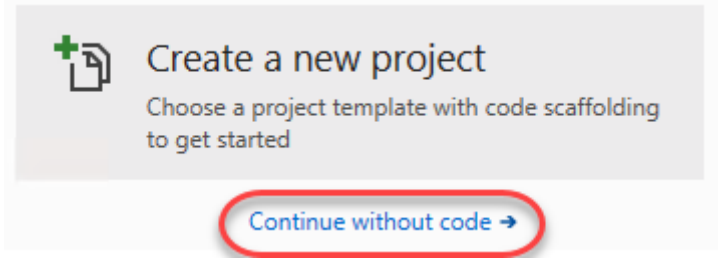


Figure 8: Visual Studio 2019, cloning a project.

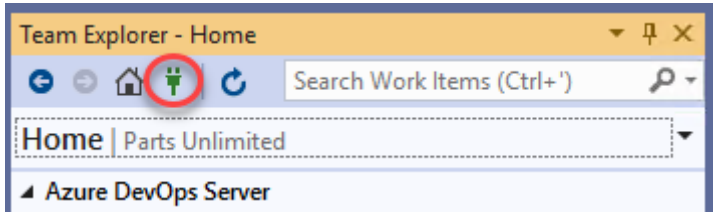


Figure 9: Show connections for the team.

Notes

.....

.....

.....

.....

.....

.....

.....

COPY PASTE)

Task 2: Cloning and configuring a GitHub project in Visual Studio

1. Open Visual Studio 2019.
2. Click "Continue without code". (*See Figure 8*)

Note that you also have the option to start the cloning experience from the welcome dialog.

3. From "Team Explorer", click the "Manage Connections" button. It is possible that you have to switch on the window "Team Explorer".

It is possible that you are already logged in to GitHub.

The next step assumes you are not logged in.

4. Make sure you see the connections part. (*See Figure 9*)
5. Under "GitHub", click "Connect …".
6. Complete the process to sign in to your GitHub account. It is possible that you are already logged in.
7. Click the "Clone" item. (*See Figure 10*). Cloning means, you are a contributor to the program. You are part of the programming team.

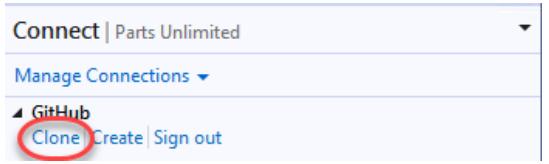


Figure 10: Clone a GitHub repository.

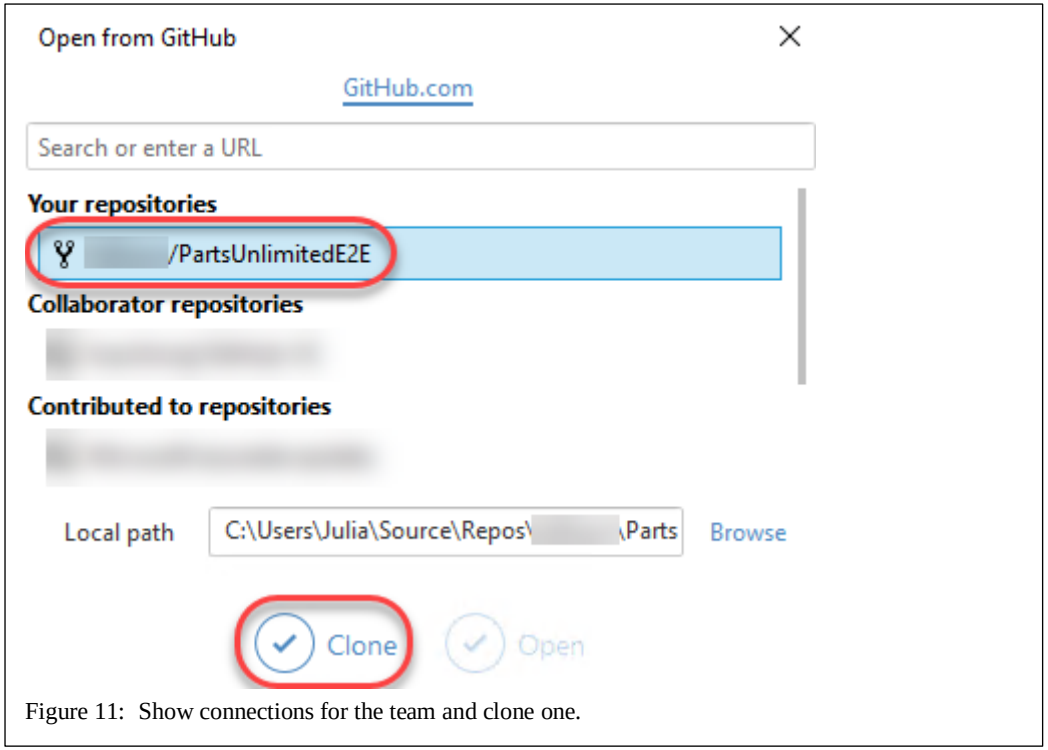


Figure 11: Show connections for the team and clone one.

8. Select the project forked earlier in task 1 and click "Clone". (*See Figure 11*)
9. Pay attention to where the project is cloned to on your computer.
10. After logging in, Team Explorer lights up with a variety of shortcuts and features to make your experience with GitHub as seamless as possible.
11. Many of the buttons are shortcuts to the GitHub portal page for this project, such as "Pulse", "Graphs", and "Wiki".
12. Click "Settings". (*See Figure 12*)

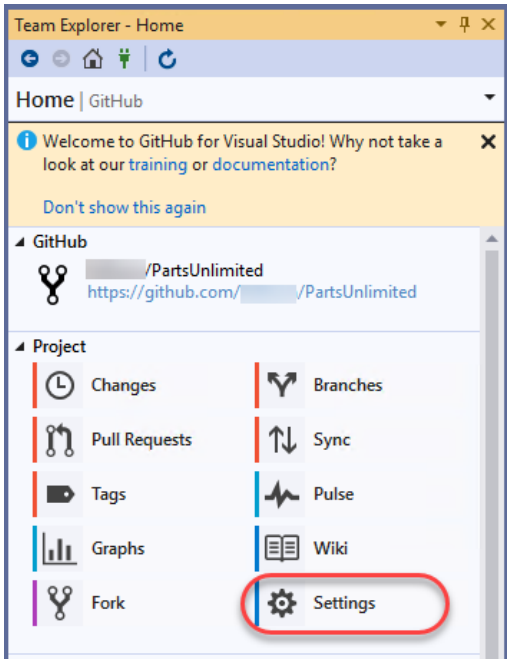


Figure 12: Go to GitHub settings.

Notes

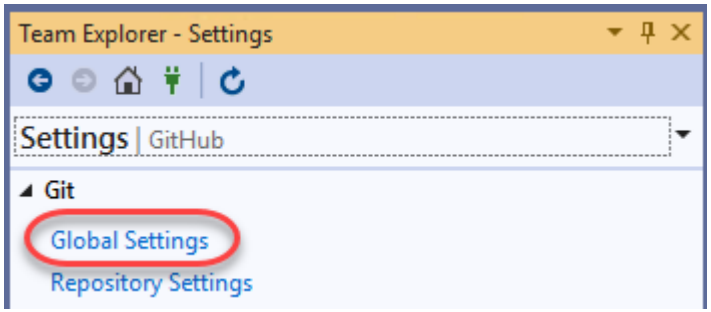


Figure 13: Show Global Settings for GitHub in Visual Studio

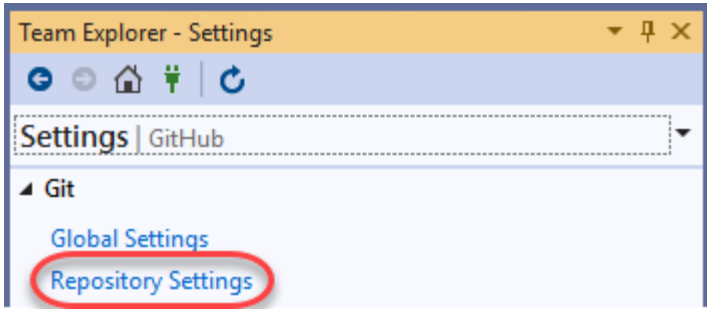


Figure 14: Show Repository Settings for GitHub in Visual Studio

Notes

13. You can configure settings at two levels.
14. Click "Global Settings". (*See Figure 13*)
15. The Global Settings view provides a way for you to set global defaults that apply to all projects.
    In this case, the "User name" and "Email" are already configured. However, you may want to change them for your instance. (*See Figure 15*)

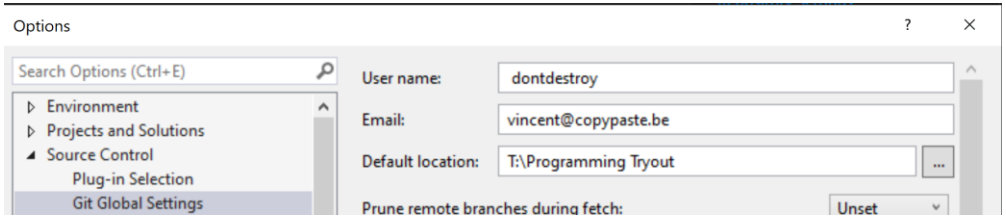


Figure 15: Show Options of the Global Settings for GitHub.

16. Click "OK" button or "Cancel" button to close the screen.
17. Click "Repository Settings". (*See Figure 14*)
18. These settings are specific to the current project. (*See Figure 16*)

Figure 16: Show Options of the Repository Settings for GitHub.

19. Click "OK" button or "Cancel" button to close the screen.
20. Click the Home button in the Team Explorer.

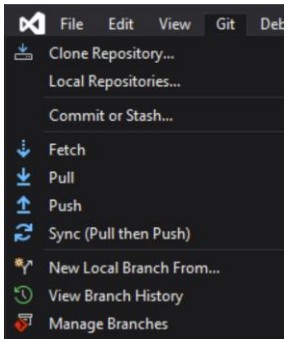


Figure 17: Having an overview on the branches.

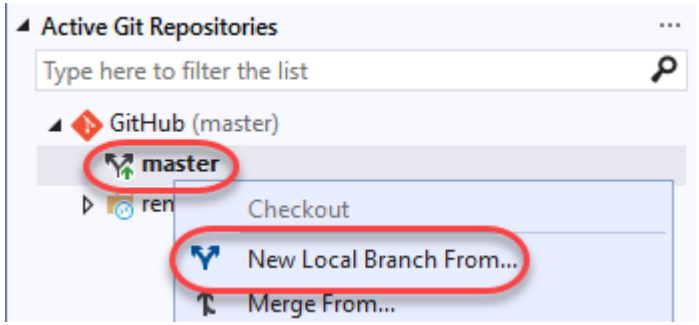


Figure 19: Creating a new Branch.

Notes

Task 3: Exploring GitHub version control integration

To get started on the work item created earlier.

1. Make sure you have your project open (the local one).
2. Click the "Git" menu, and then "Manage Branches". (*See Figure 17*)
3. The work will be done on a separate branch and merged in after a review.
4. Right-click the "master" branch and select "New Local Branch From". (*See Figure 19*)
5. You see the "Create a new branch" dialog. (*See Figure 18*)

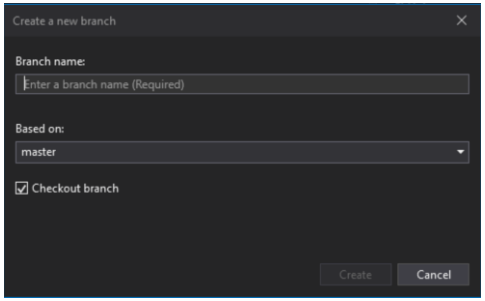
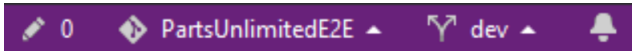


Figure 18: Creating a new Branch dialog.

6. Enter a good Branch Name. (in the example it is "dev").
7. Make sure that your branch is based on the "master".
8. Also check out your branch by checking "Checkout branch".
9. Set the name to "dev" and click "Create".
10. The new branch will be checked out after creation.

Note that you can see the current branch and perform common options using the button at the bottom of the window



COPY PASTE

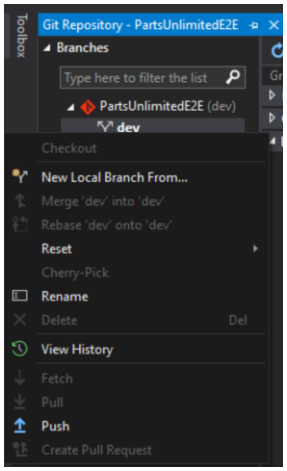


Figure 20: Pushing the new Branch towards GitHub Server.

```
20      <div class="pull-left logo">
21          <a class="block logo-image" href="@Url.Action("Index", "Home")">
22              <img src="/images/unlimited_logo.png"/>
23          </a>
24          <a class="block" href="@Url.Action("Index", "Home")">
25              <h1>Parts Unlimited v2.0</h1>
26              <h2>a Fabrikam subsidiary</h2>
27          </a>
```

Figure 21: Changing some text / code / information.

Notes

.....

.....

.....

.....

.....

.....

.....

COPY PASTE)

11. Right-click the "dev" branch in the Git Repository screen and select "Push". This will push the locally created branch to the server. (*See Figure 20*)

From here on, we can change the code.

12. From Solution Explorer, search for "_layout" and open the _Layout.cshtml from the PartsUnlimitedWebsite project. (*See Figure 22*)

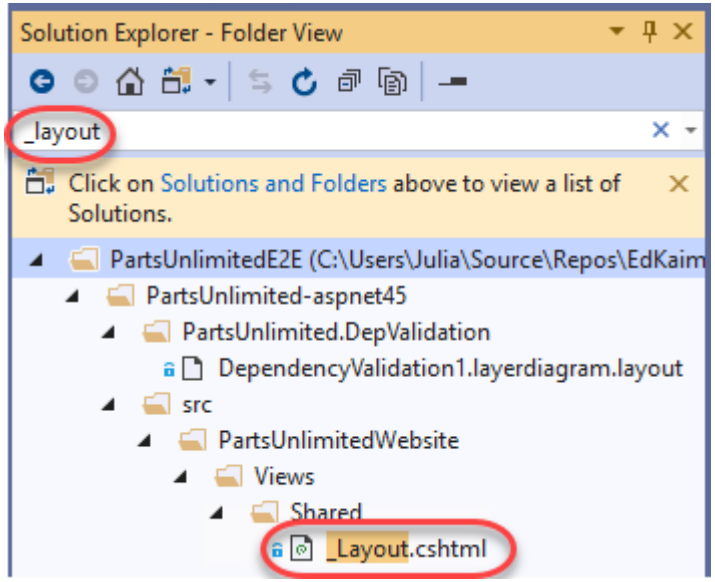


Figure 22: Select a file and change something in it.

13. Add "v2.0" to the h1 tag text and Save the file. (*See Figure 21*)

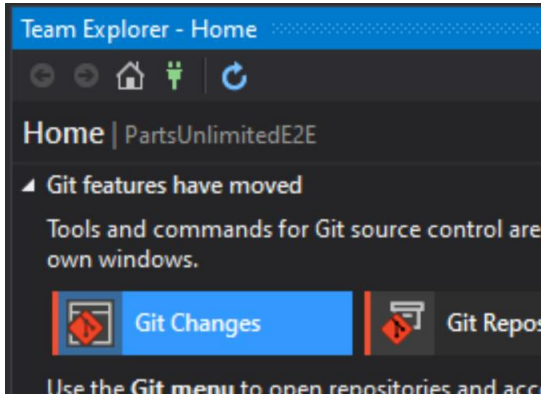


Figure 23: Looking at the changes that are done.

14. In Team Explorer, switch to “Git Changes”. (*See Figure 23*)
15. All the changes (only one) will be shown. (*See Figure 24*)

You can double click that file, to see the differences caused by changes.

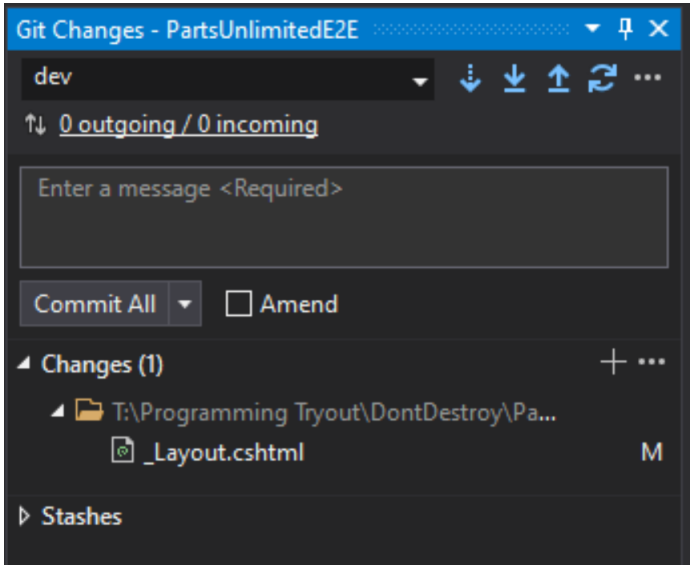


Figure 24: Commit the changes and Syncronize.

16. Enter a commit message of “Updated to v2.0” and select “Commit All” > “Commit All and Sync”.
17. This will commit your change and push it to the GitHub server.

Notes

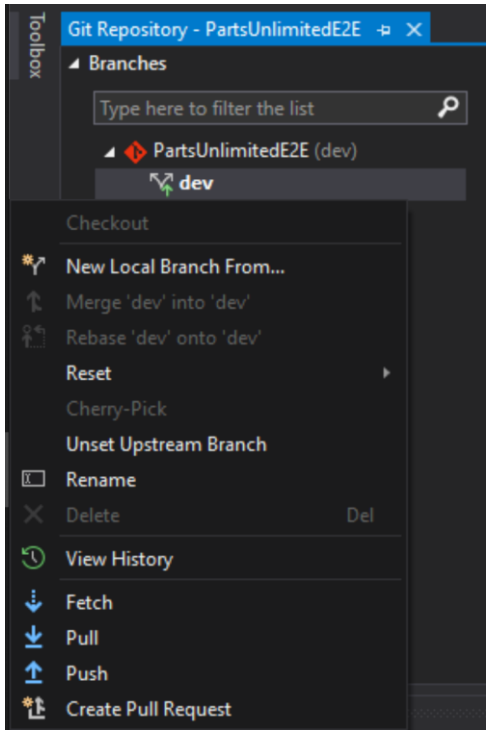


Figure 25: Create Pull Requests.

Task 4: Exploring GitHub pull request integration

Although the dev branch has been updated with the necessary change, it still needs to work its way back in to the master. This can be done with a pull request.

1. Click the Home button in Team Explorer.
2. Click "Git Repository".
3. Right-click the "dev" branch and select "Create Pull Request". (*See Figure 25*)
4. You are going to GitHub and you see the repository of Microsoft.

Note that it will default to the Microsoft project, which you do not want to use.

Make sure you have your local Master Respository.

5. Set the branch to merge into to master from your project. (*See Figure 26*)

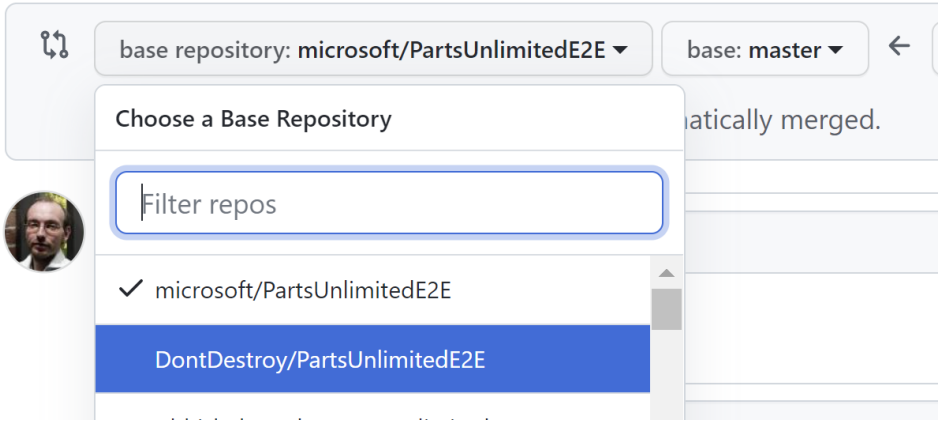


Figure 26: Select the correct base repository (your local one).

Notes

COPY PASTE)

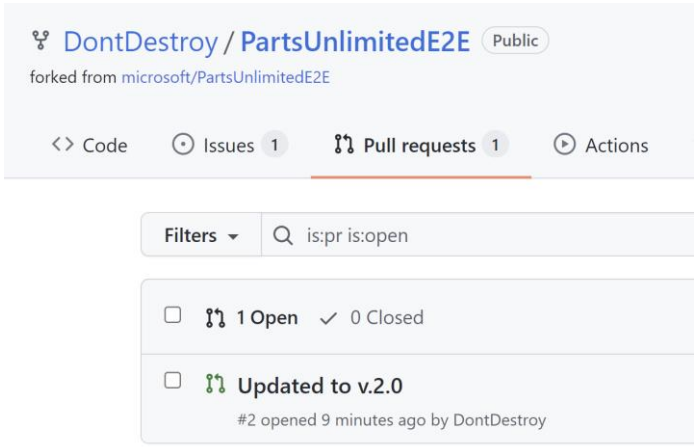


Figure 27: Having an overview of the pull requests.

6. Set the comment to “Fixes #1.”.

Note that you may need to replace the #1 with the ID created earlier if it were different. This should be the issue number you just have created.

By tagging the pull request with the issue ID, you can automate closing the issue later on when the request is merged.

7. Click “Create pull request”.
8. In your local repository, you can see the pull requests. Click on “Pull requests”. (*SEE FIGURE 27*)
9. Click the created pull request to open it and see the details.

The pull request view includes all the information you need to review changes and make comments.

10. In the tab “Files Changed”, you can see the differences. (*SEE FIGURE 28*) The diff viewer makes it easy to understand what changes were made and where.

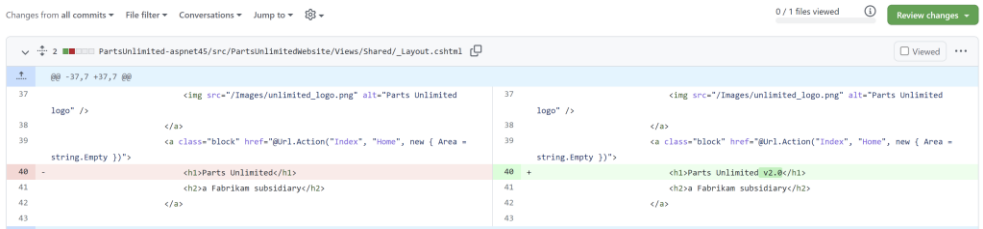


Figure 28: See the differences by the change.

11. Alternative. This can also be done in Visual Studio. You can see the same stuff in Visual Studio. By right clicking the commit and select “View Commit Details”.

Notes

COPY PASTE)

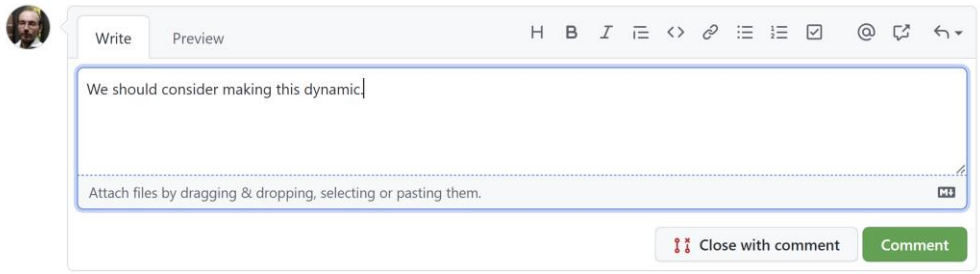


Figure 29: Typing a comment.

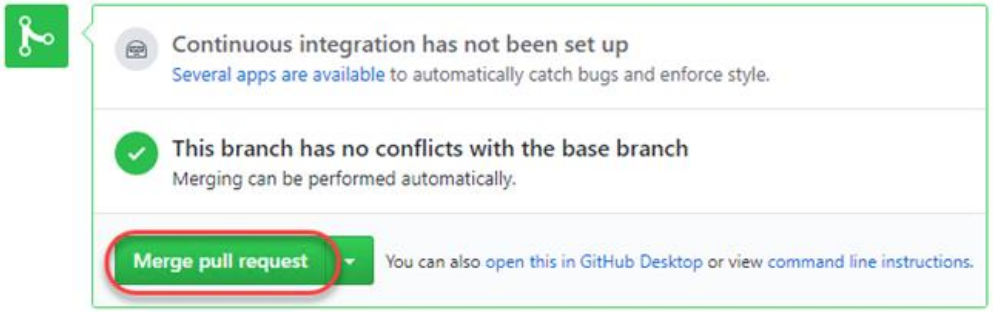


Figure 30: Merge pull request.

Notes

COPY PASTE)

12. You can also leave line-level comments. Go to the "Write" tab en type some comment.
13. Type "We should consider making this dynamic". (*SEE FIGURE 29*)
14. Click "Comment" button to save.
15. Your review is now visible as part of the pull request.
16. Go to the "Write" tab en type some comment.
17. Type "It looks good for now".
18. Click "Comment" button to save.

Others can comment or review the changes as well

19. Click Merge pull request. (*SEE FIGURE 30*)
20. Confirm the merge. (*SEE FIGURE 31*)

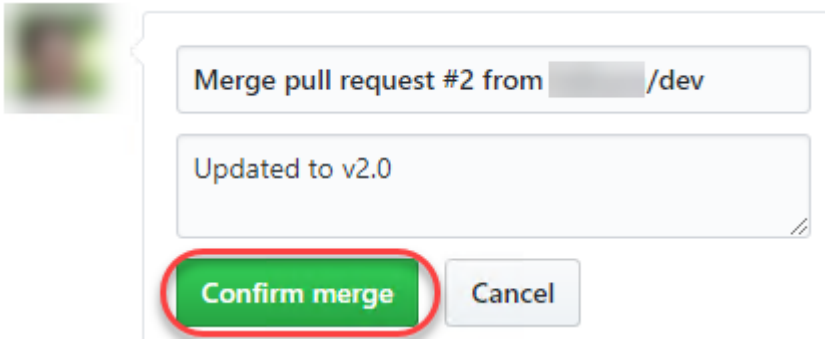


Figure 31: Confirm the merge.

21. Navigate back to the Issues tab.
22. Note that the issue created earlier has been closed now that the pull request was approved.