

the trusted technology learning source



Home > Articles

Open Source: GitHub for .NET Developers, Part 2: Collaborating and Contributing



By [Alessandro Del Sole](#)

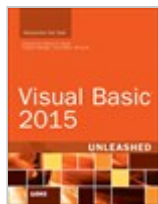
Sep 14, 2016

 [Print](#)  [Share This](#)


Page 1 of 1

In Part 2 of this series, Alessandro Del Sole explains how other developers, either within a team or from the outside developer community, can collaborate on an open source project hosted on GitHub.

From the author of



[Visual Basic 2015 Unleashed](#)

[Learn More](#)  [Buy](#)

Working in Teams

In Part 1 of this series, I explained what open source and GitHub represent today for developers working with Microsoft technologies. I also presented an overview of how to create an open source repository on GitHub and work with the repository on your own—even across different machines.

One of the biggest benefits of hosting open source projects on GitHub is that it allows multiple developers to work on a project as a real development team. In this context, every developer on the project is called a *collaborator*. When you create a repository, you automatically have administration privileges for it, and you can add other developers to your project. (Of course, every collaborator must be registered on GitHub.)

To add collaborators, open the repository's home page and select **Settings**. On the Settings page, click **Collaborators**, and then enter the collaborator's name in the search box. When ready, click **Add collaborator**. In the [Figure 1](#) example, I'm adding an alternate GitHub account I own for demonstration purposes.

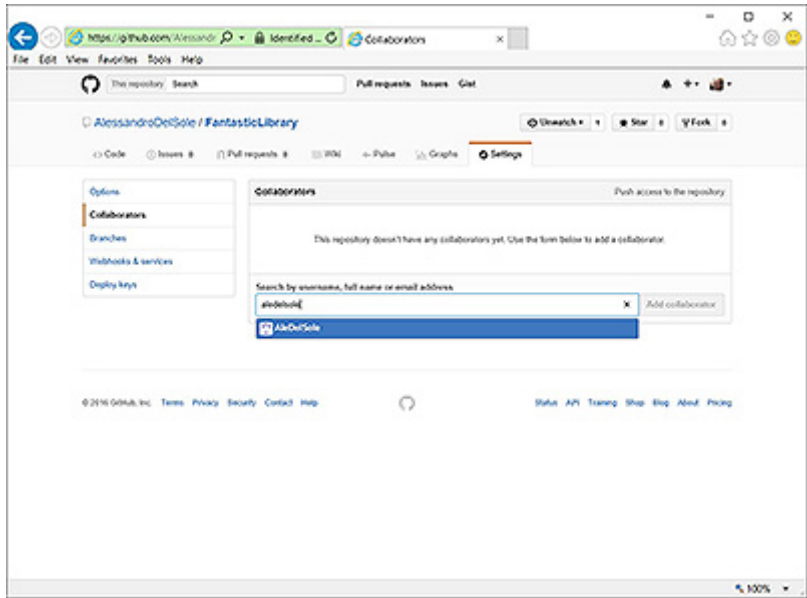


Figure 1 Adding collaborators.

NOTE

The collaborator will be notified via email after you have added that person to the open source project.

By default, every collaborator has read/write permissions for a personal repository (like the one used in this article series). For an organizational repository, the repository's administrator has more granularity in assigning specific permissions.

Collaboration: Working as a Team

Collaborators on the development team will need to clone the repository on their machines, and then every commit that a collaborator pushes to GitHub will be merged into the main repository. In Part 1, I briefly mentioned the concepts behind cloning a repository; now let's look at the procedure.


Collaborators launch Visual Studio 2015 and click the **Manage Connections** button in Team Explorer. Then they enter their GitHub credentials (if required), and finally click **Clone**. As [Figure 2](#) shows, when cloning a repository, collaborators get a list of repositories they own as well as a

Related Resources

[Store](#)

[Articles](#)


[Blogs](#)



[Cloud Native Go: Building Web Applications and Microservices for the Cloud with Go and React](#)

By [Kevin Hoffman](#), [Dan Nemeth](#)


Book \$31.99



[Effective C# \(Covers C# 6.0\). \(includes Content Update Program\): 50 Specific Ways to Improve Your C#, 3rd Edition](#)

By [Bill Wagner](#)

Book \$35.99



[ASP.NET Core Application Development: Building an application in four sprints](#)

By [James Chambers](#), [David Paquette](#), [Simon Timms](#)

Book \$35.99

[See All Related Store Items](#)

list of repositories in which they're involved.

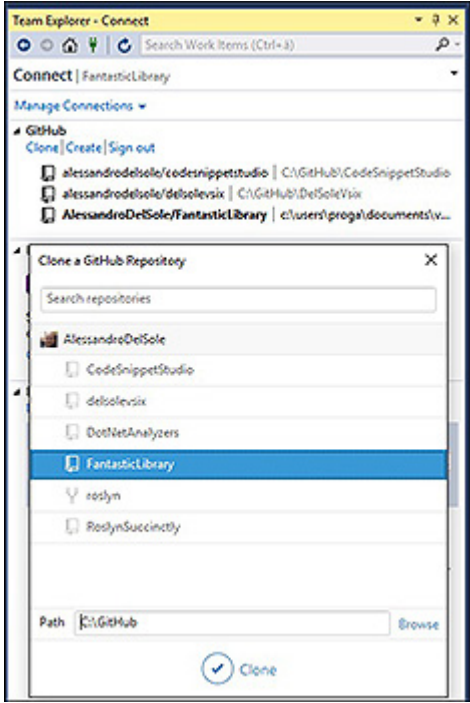


Figure 2 Cloning a repository.

NOTE

You can also start cloning a repository from the GitHub website, using the **Clone** button in the repository's home page. With this approach, you'll be prompted to choose between Visual Studio and the [GitHub Desktop](#) application (which synchronizes repositories but doesn't allow editing) to finish cloning the repository locally. So you might want to begin directly in Visual Studio.

At this point, because the repository is under source control, every edit that a collaborator commits to GitHub is merged into the main repository. The repository owner can see all edits per commit on the GitHub website, and can reject any commit by reverting the repository to a previous commit.

As I discussed in Part 1, edits made in the source code are shown in the Visual Studio 2015 Enterprise editor via the CodeLens feature, which makes it really easy for project managers and administrators to understand a team member's work on a piece of code.

Managing Activities and Bugs: The Concept of Issues

Applications may have bugs, ongoing projects may need features to be implemented, and the project manager might want to assign tasks to collaborators. GitHub offers a very convenient way of tracking such problems and tasks, which it terms *issues*. Basically, an issue represents something to do, such as fixing a bug or implementing a feature. You access the project's issues by clicking **Issues > New Issue**. For each issue, you provide a title and a description, as shown in [Figure 3](#). The text editor supports the Markdown styling syntax. Notice that the **Assignee** hyperlink allows you to assign the issue to a collaborator or yourself.

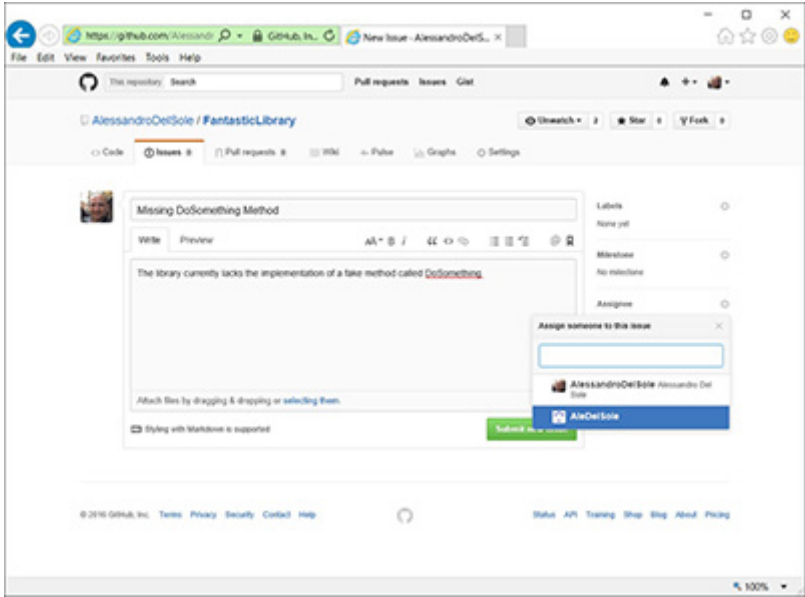


Figure 3 Creating a new issue.

When you're finished editing the issue, click **Submit new issue**. GitHub shows a summary for the issue. At any time, you can simply click the **Issues** tab in the repository's home page to get the list of existing issues (see [Figure 4](#)).

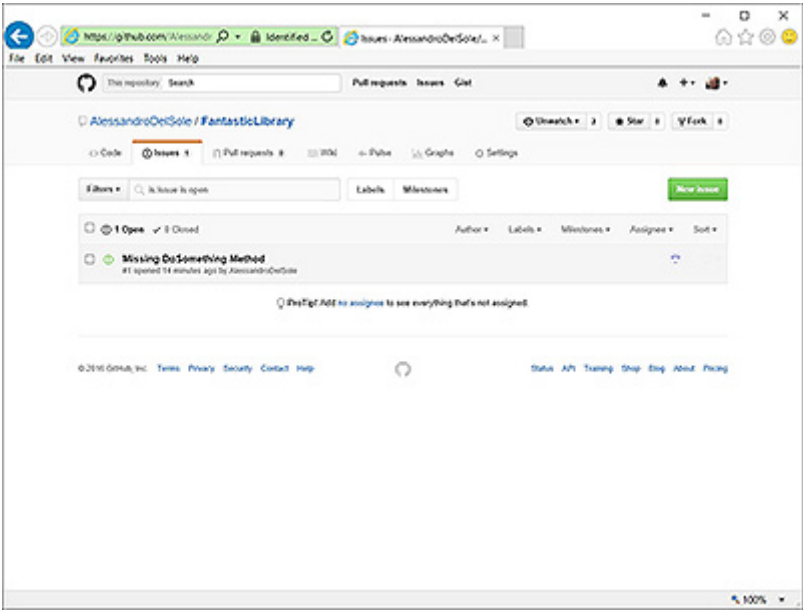


Figure 4 Showing the list of active issues.

By default, GitHub shows the list of active issues, rather than all issues. If you want to include closed issues, clear the filter text in the **Filters** box and press Enter. You can click an issue in the list to provide edits, or just to close it after the related task has been completed. [Figure 5](#) shows how a closed issue looks.

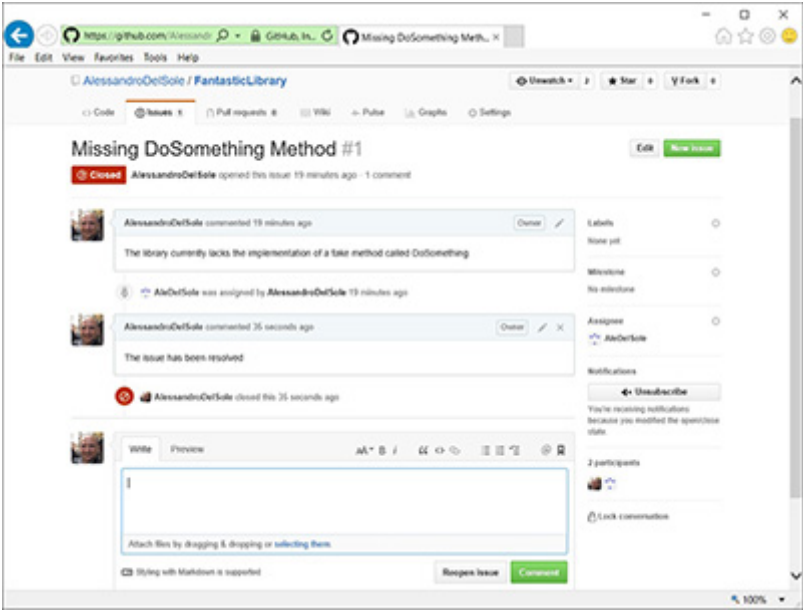


Figure 5 Closing an issue.

Following is a short but important list of considerations about issues:

- It's good practice to leave a comment when you close an issue, explaining how the issue has been resolved. This information is useful for project managers, but also for users who download, reuse, or maintain the code.
- An issue can be reopened if the problem hasn't been solved as expected, or if the feature doesn't work as expected.
- Because an open source repository can be downloaded by anyone, any registered GitHub user can submit an issue.

Issues offer a reliable way of tracking problems and activities in a repository.

Contributions from the Developer Community

Most of GitHub's success depends on the possibility of accepting contributions from the developer community worldwide. In fact, any registered user can potentially contribute to any repository. When you work alone or with a development team, you commit your new versions of the code to the repository. But how can developers who are not on your development team provide contributions? The following sections explain how other developers can work with your repositories.

Cloning Repositories with Forks

As the owner of an open source repository, you might want to accept contributions from the rest of the development community, in addition to your project collaborators. But you still need to maintain control over code submissions and contributions. For this purpose, GitHub provides an option to clone a repository that prevents users from committing changes directly, so that the project owner(s) can review all changes before they're merged into the repository. This is accomplished by *forking* the repository.

GitHub's Fork tool basically creates a copy of the specified repository to the developer's account. When the developer makes some edits to the code, changes are submitted to the *cloned* repository—not the original one. At this point, if the developer introduces improvements, fixes bugs, or simply makes changes that might be important to improve the project, she needs to submit a *pull request*. A pull request can contain code, suggestions, ideas, and so on. Every time someone sends a pull request, the owner of the original repository receives a notification about the request and can decide to accept or reject the requested changes.

To understand how this system works, let's access GitHub with a different account and search for the FantasticLibrary repository's URL (for this example, <https://GitHub.com/AlessandroDeSole/FantasticLibrary>). When ready, click **Fork** (see [Figure 6](#)).

NOTE

The URL above is for demonstration purposes only; it doesn't really exist. You'll need to work with an existing repository.

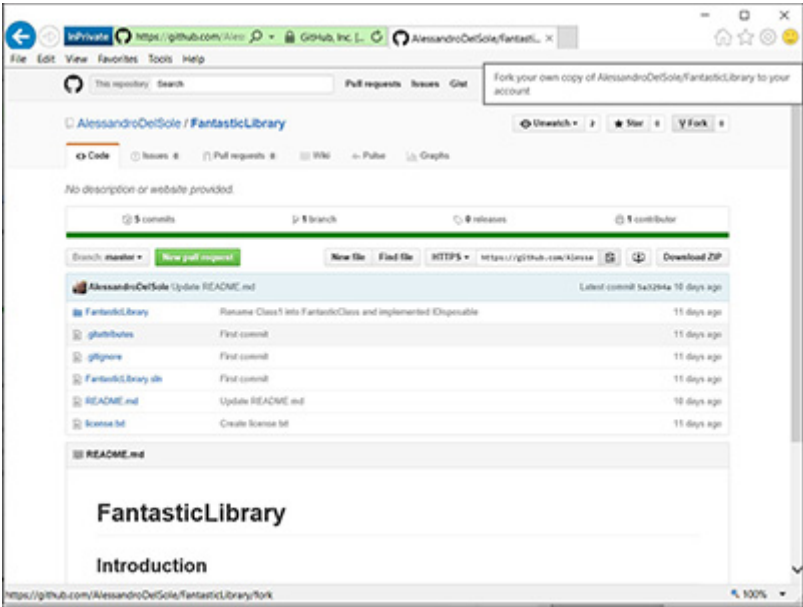


Figure 6 Preparing to create a fork.

After a few seconds, the account will receive an exact copy of the original repository, as demonstrated by the **Forked from..** wording below the repository name (see [Figure 7](#)).

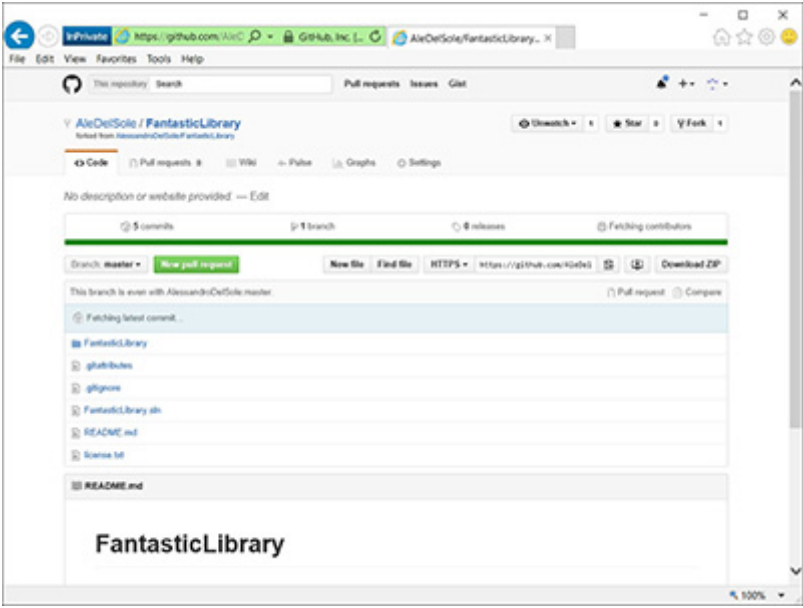


Figure 7 A fork has been created from the original repository.

Now that the current account has a copy of the original repository, changes can be made and submitted to this copy. You can use either Visual Studio 2015 or the GitHub editors to make edits and submit changes. For example, in [Figure 8](#) I've edited the Class1.cs code file to add a class definition for a type called IncredibleClass, which inherits from FantasticClass.

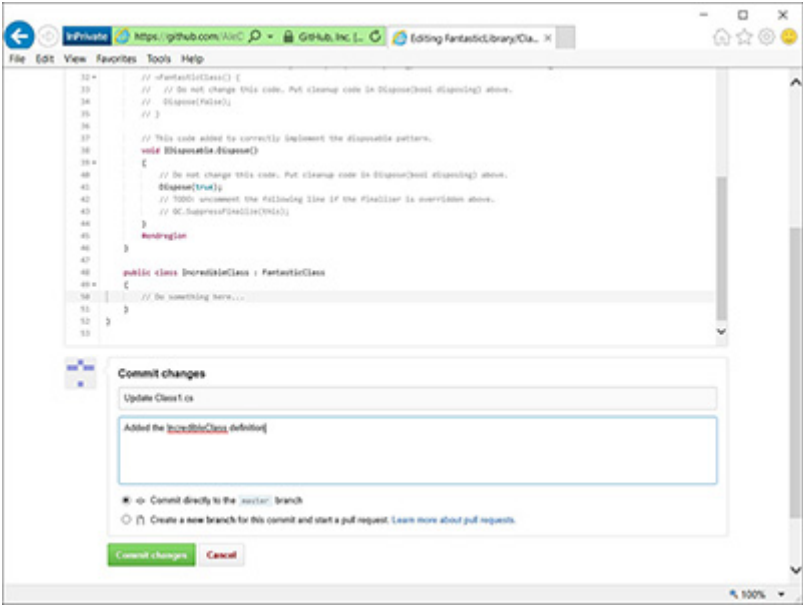


Figure 8 Making edits to a forked repository.

When you're finished editing, enter a description for your edits and click **Commit changes**. This command sends a new version of the code file to the forked repository. Commits work the same way with forks as they do with projects you own, maintaining local and remote repositories. If you go back to the list of files and click the commit description, you'll be able to compare changes, as shown in [Figure 9](#).

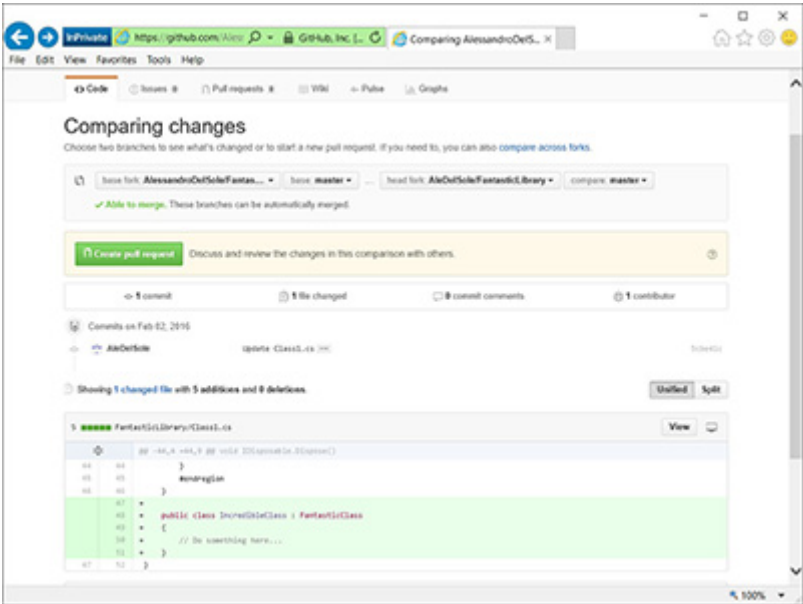


Figure 9 Comparing commits in a repository.

NOTE

Comparing commits works the same way whether you're coding solo or on a development team.

Sharing Your Contributions: Sending Pull Requests

Suppose you want to ask the owner of the repository you cloned to review and merge your contribution. You need to create a pull request, so click the **Create pull request** button. Now you can edit your pull request, providing a proper description, as shown in [Figure 10](#).

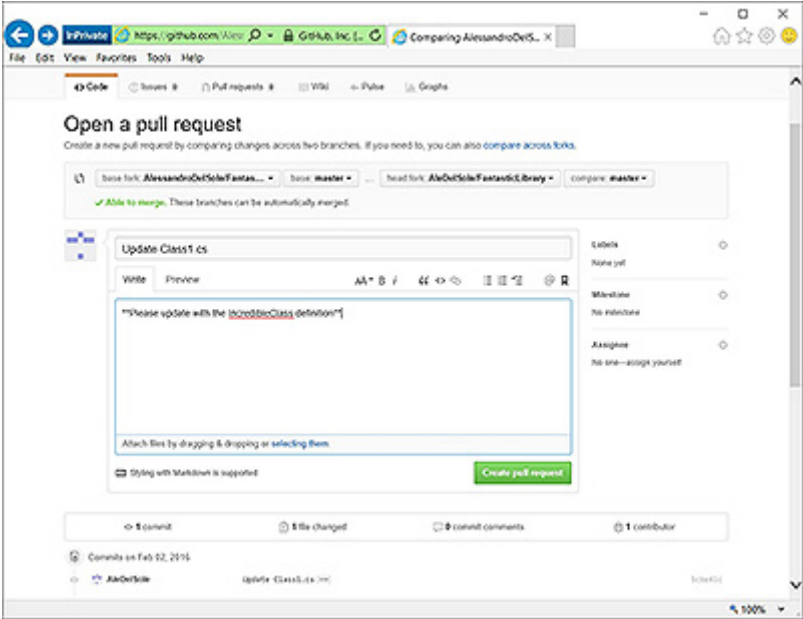


Figure 10 Preparing a pull request.

Notice that the request's title contains the commit name and cannot be modified. When ready, click **Create pull request**. After a few seconds, GitHub shows a summary page for your pull requests. Concurrently, the owner of the original repository you forked receives a notification of the pull request, and has an option to review and approve (or reject) your pull request. If the owner approves your pull request, the code you submitted is merged into the code in the original repository. [Figure 11](#) shows the kind of review page the owner sees before merging a pull request.

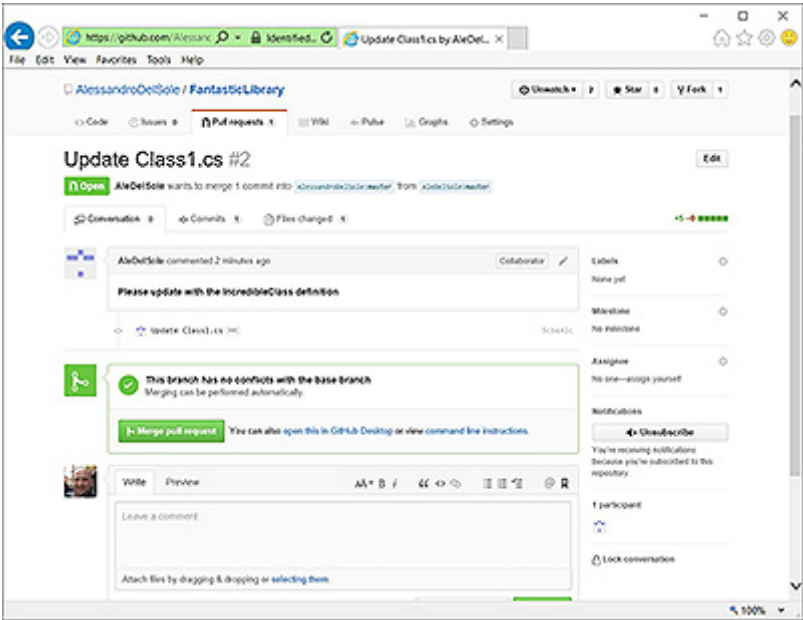


Figure 11 The owner of the repository can review a pull request.

If your pull request is acceptable, the owner clicks **Merge pull request**. After a few seconds, a page shows the result of merging the code. It also provides an option to revert to the previous state (that is, the latest commit before merging). [Figure 12](#) demonstrates this option.

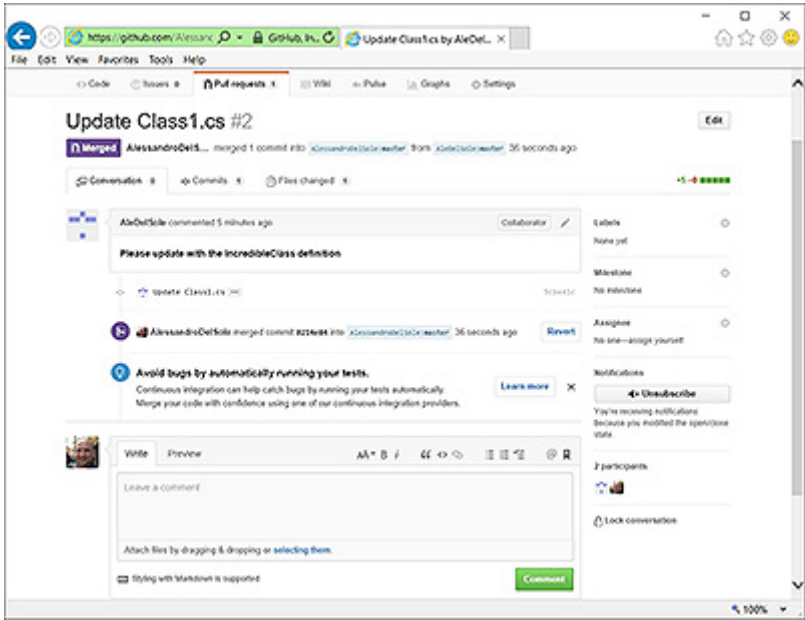


Figure 12 Reviewing the merge operation.

Pull requests like these are how Microsoft accepts contributions from the developer community worldwide for its open source projects, including the Visual Basic and C# compilers. If you want to contribute to an open source technology that Microsoft produces, you can fork the repository, edit the code, and send a pull request with your new or revised features. Just remember to follow Microsoft's strict [rules](#) when you submit your contributions.

Summary

GitHub provides the proper infrastructure to support both team collaboration and contribution from the developer community. Whether you're invited to join a team or you want to send your contributions, you can clone a repository, improve the code, send it back for review, and use issues to track your work. As a developer working with the Microsoft technologies, you can leverage the Visual Studio 2015 integrated tooling to work against GitHub repositories from within the IDE, so that (in most cases) you'll just focus on your code.

[+ Share This](#) [Save To Your Account](#)