**Azure DevOps Guide for Visual Studio Projects**

Contents

[**Setting up Azure DevOps** 1](#_Toc40348822)

[**Code development & Reviews** 7](#_Toc40348823)

[**Dealing with Merge conflicts** 10](#_Toc40348824)

# **Setting up Azure DevOps**

**Prerequisites** –

Visual Studio 2017

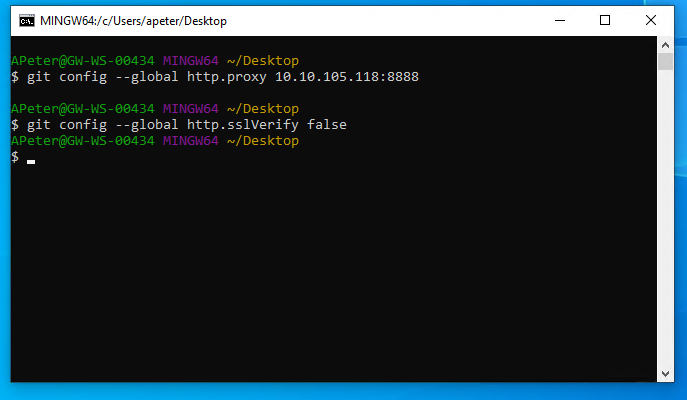
Git for Windows (<https://git-scm.com/>) or from “\\storage-server\Development Drive\Development Tools\git” and install using the default options.

Link to Glasswall core azure page - <https://dev.azure.com/glasswall/Glasswall%20Core>

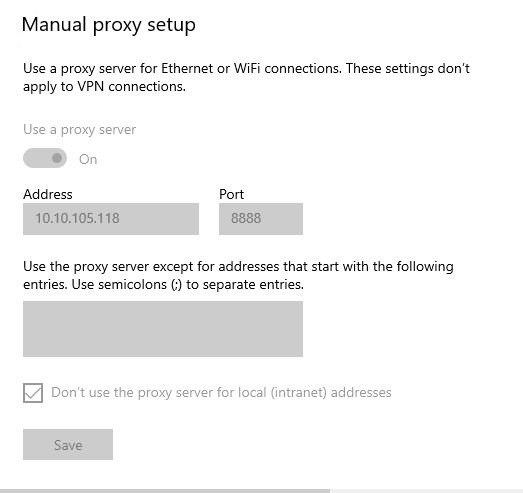
Link to official documentation - <https://docs.microsoft.com/en-us/azure/devops/repos/git/gitquickstart?view=azure-devops&tabs=visual-studio>

**Steps -**

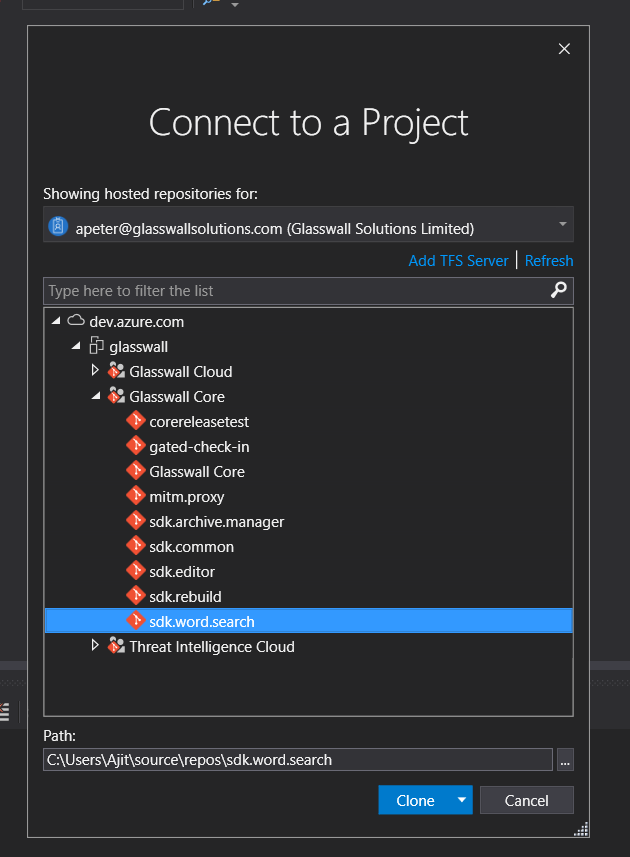
1. Open Git Bash and add the following settings. Ensure that the ip address used is the same as shown in your proxy settings.



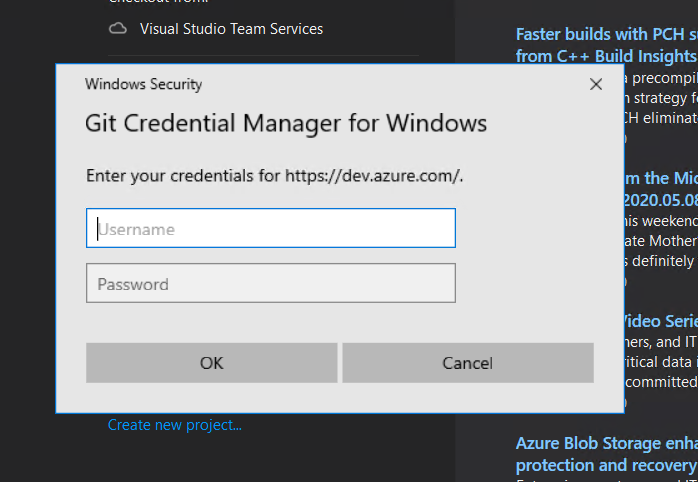
The ip address to be used can be viewed by opening “proxy settings” from windows start menu. The address shown would be your proxy ip address. Below is an image of how a proxy settings page would look like.



1. From Visual Studio 2017 connect to project using your work email account. This would present a screen as shown below and clone the repo to the local machine. The example shown here is for the word search project.



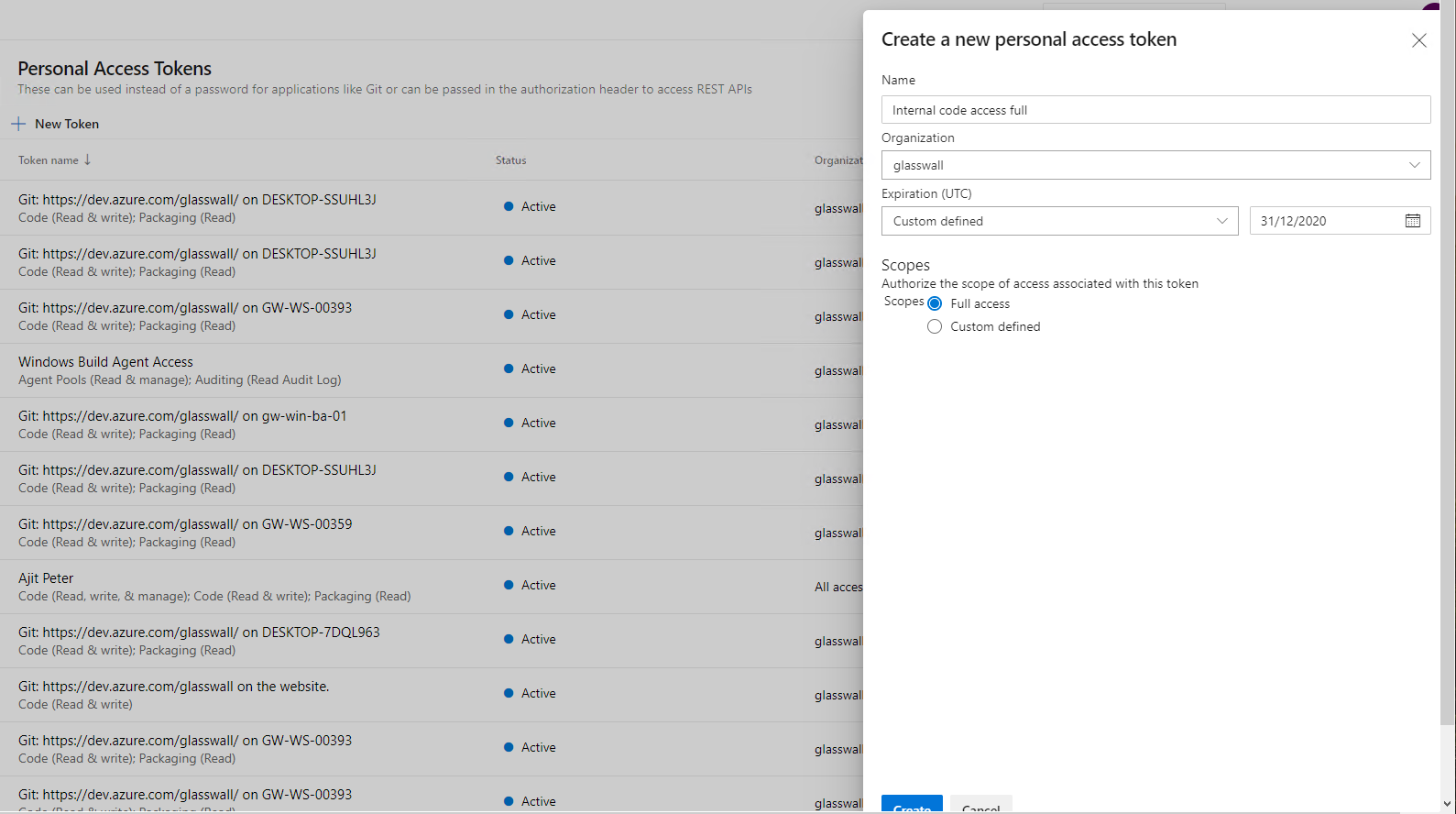
1. Once it is connected clone the repository that you intend to work on. This would bring a screen like this. The username is your work email address and the password is a personal access token(PAT) that you would have to create.



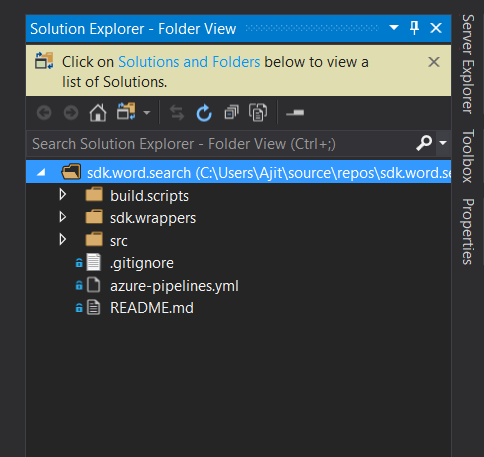
**How to create a Personal Access Token?**

Click on the following link <https://dev.azure.com/glasswall/_usersSettings/tokens>

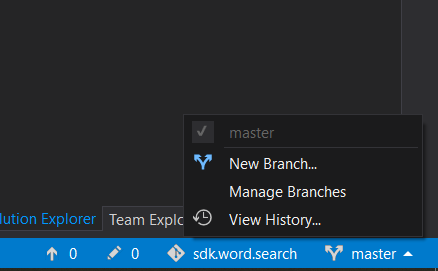
and then click on create new token. Create a token for a year with full access as shown below in the image and then name token. Keep a copy of token created and you would be able to use it as password while cloning the repo.



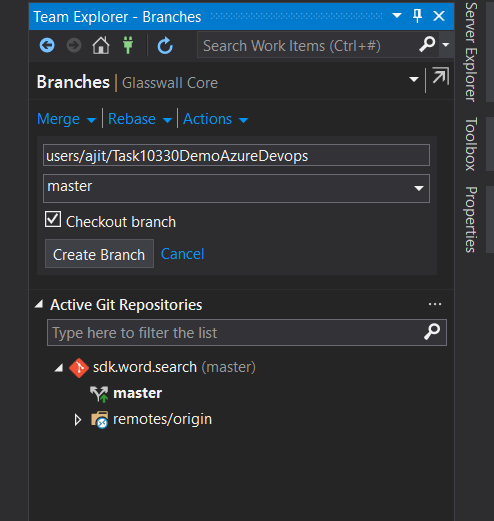
1. After the repository is successfully cloned. Click on Solutions and folder and select the solution to work on.

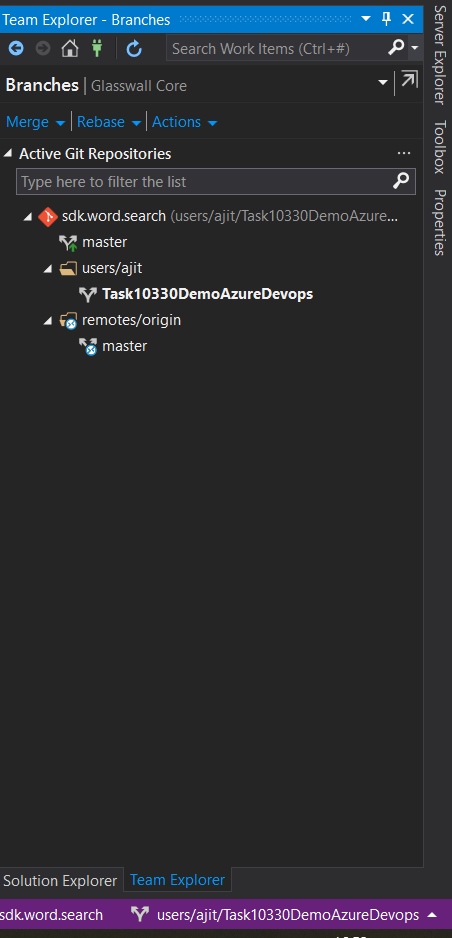


1. Once the “master” branch has been loaded the next step is to create a branch to work on. Click on “master” in the bottom right corner and select “New Branch”.



1. The branch naming convention that we are going to use would be “users/name/workitemdescription”. An example would be as follows “users/ajit/Task10330DemoAzureDevops”.

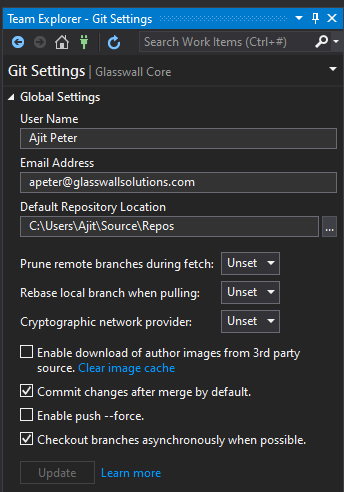


1. Once the creation of a local branch is successful, we are ready start working on the task at hand. Information on the active branch can be viewed in the bottom right corner.

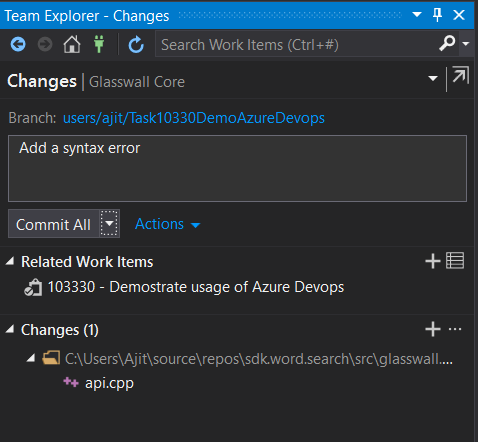
# **Code development & Reviews**

**Steps -**

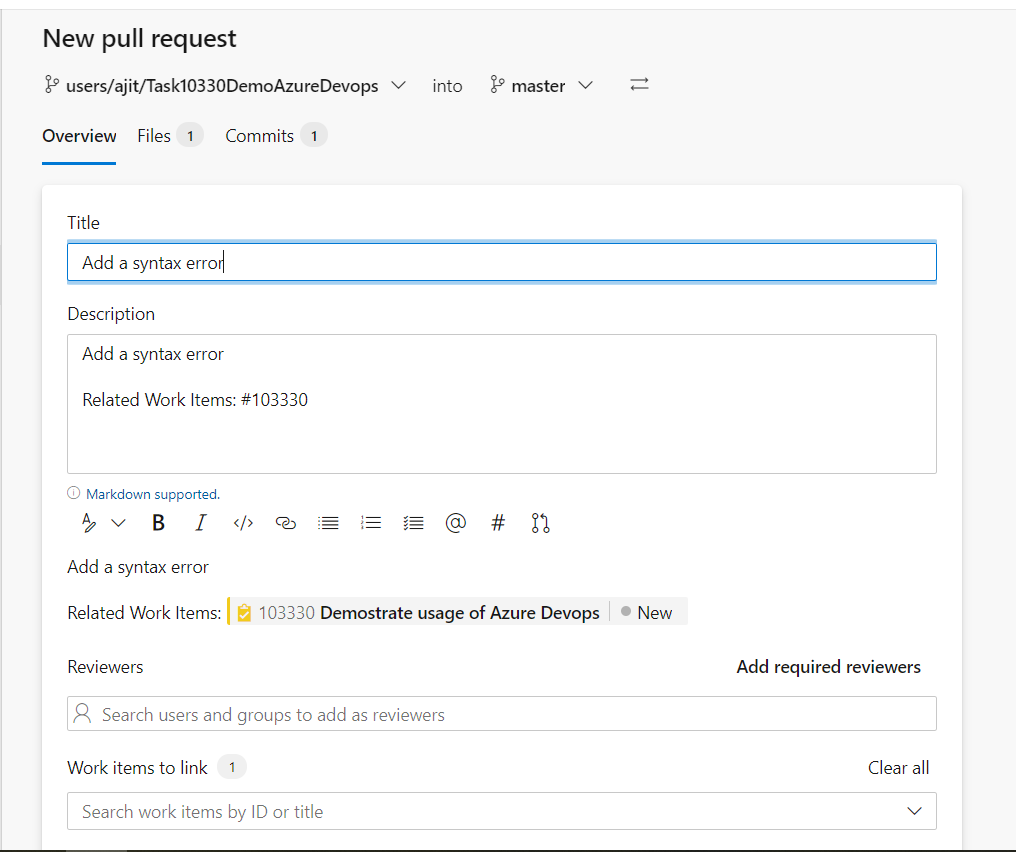
1. Setting up git repository global settings is required in order to push to the repository. It can be accessed from “Settings” in Team Explorer. Click on “Global Settings”. User Name and Email has to be set.



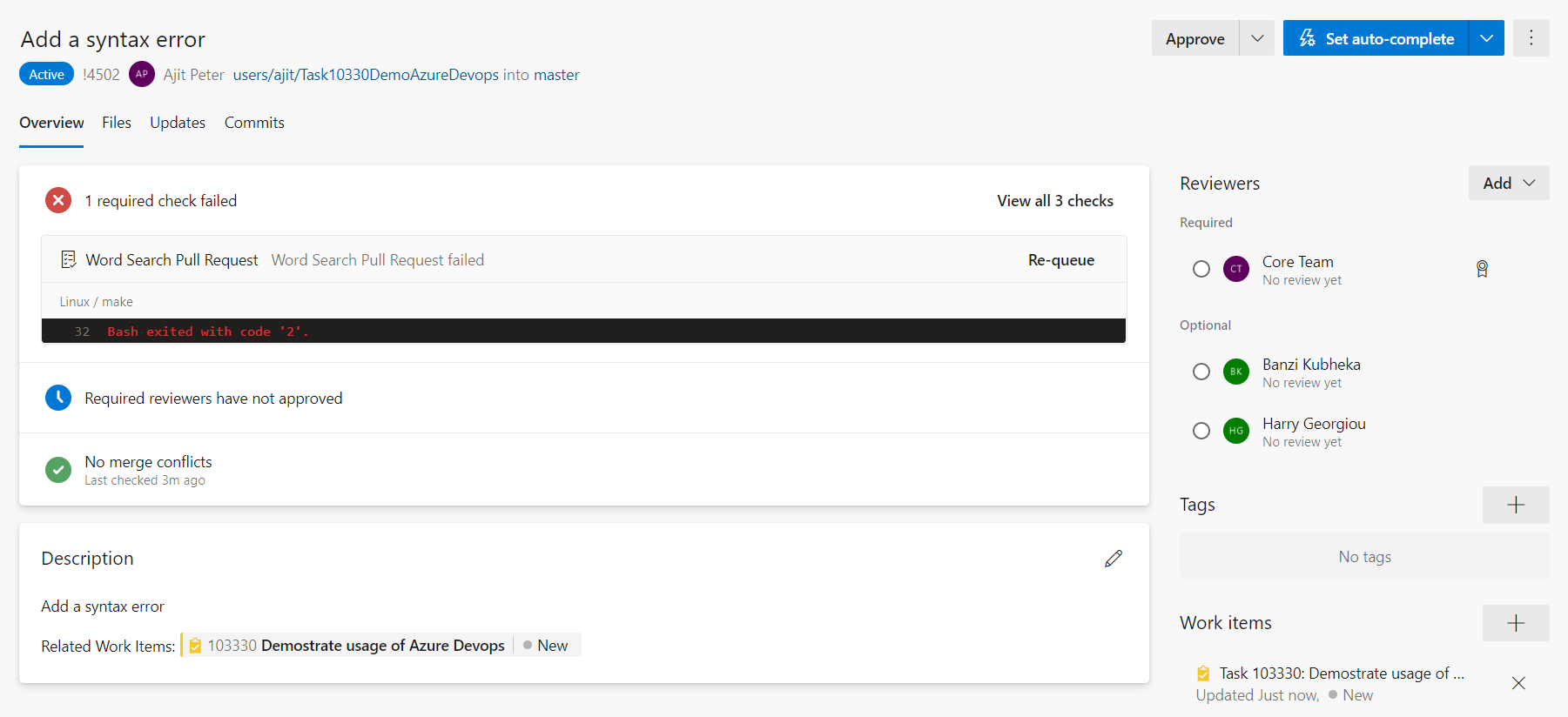
1. Make the required changes to the source file and click on “Changes” in Team Explorer window which would show all the pending changes.
2. Add a commit message and add the associated work item to change that is ready to be committed. Select “Commit All and Push” from the “Commit All” dropdown.



1. Before creating a pull request in order to make the local branch up to date with the “master” branch pull the latest change on to the local “master” branch and then merge them on your branch. An example of how to merge the latest version of “master” to your local branch is shown [here](#MergeScreenshot). The changes are now committed to the remote branch in Azure DevOps repository and it now ready to be submitted for pull request(review) so that the changes can be merged into “master” branch.
2. Pull requests can be created from Team Explorer window. Click on new Pull requests and that would take you to a similar web page. It adds reviewers from the Core Team group by default. Reviewers can also be added from web page. The review must be approved by two reviewers for the code to be merged to master.



1. Creating a pull request takes you to this page which indicates the status of the pull request. It also indicates if there are any errors in the code or associated tests.

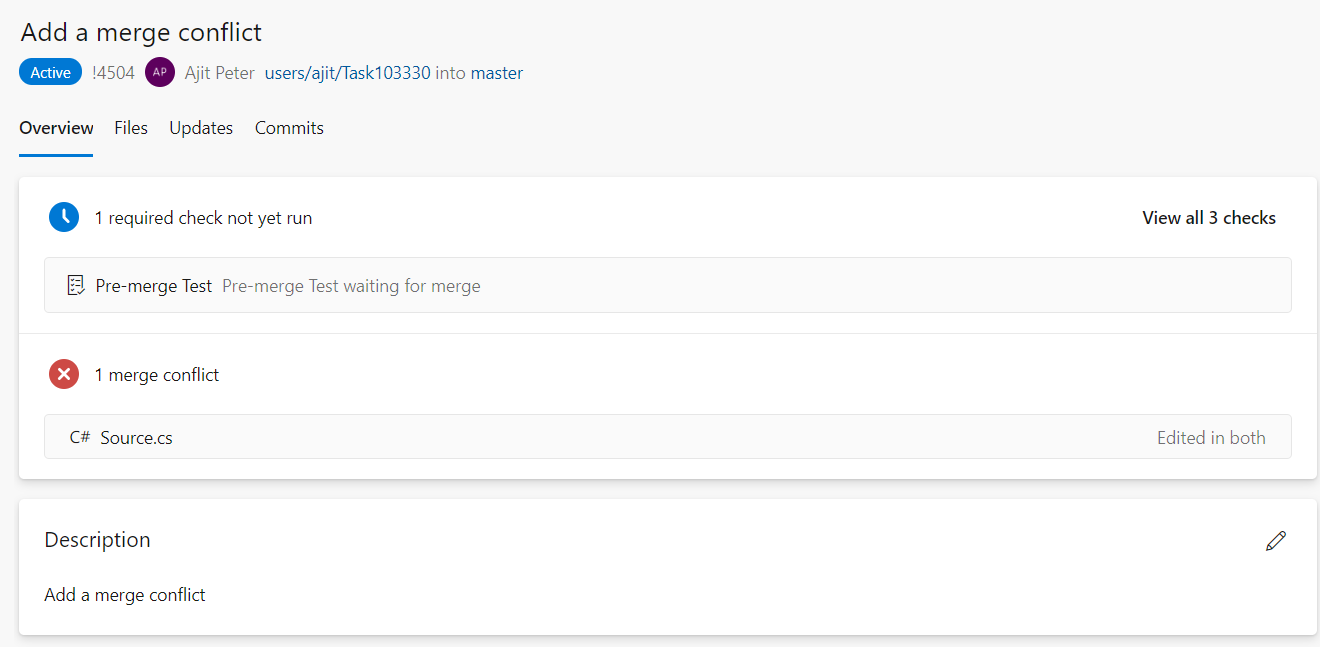


1. Once all the mandatory conditions are met the code is ready to be merged to “master” branch and can be done from the above page.

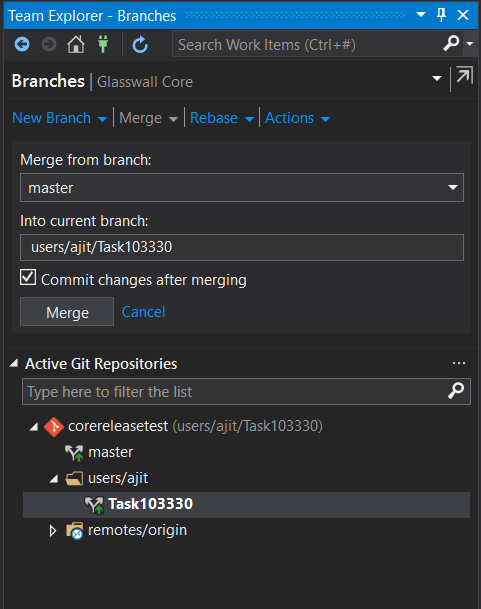
# **Dealing with Merge conflicts**

**Steps -**

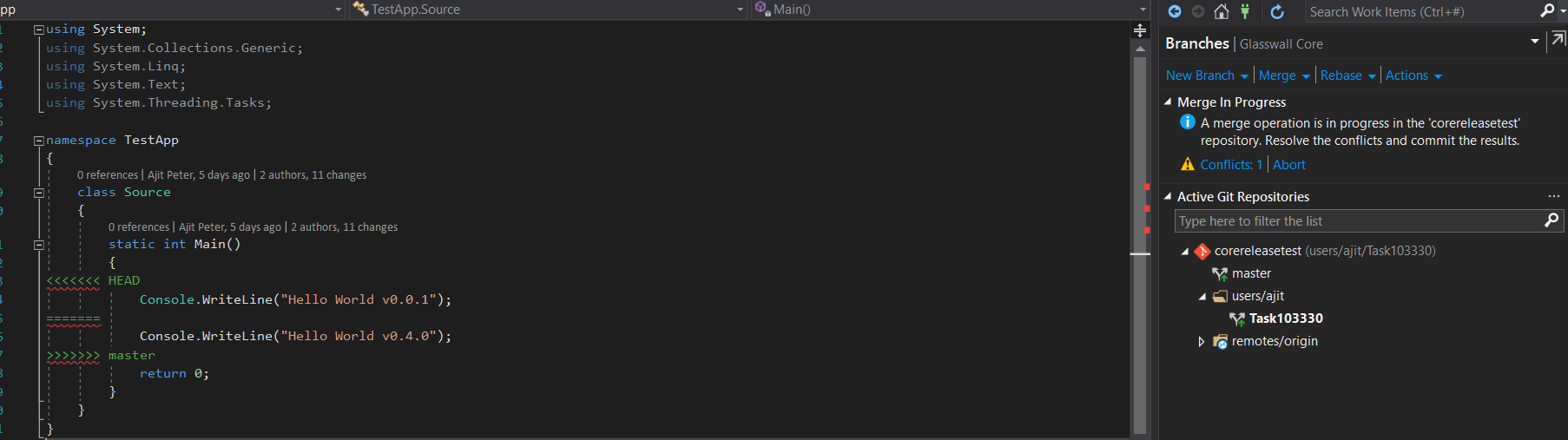
1. Once the changes are ready to be merged into “master” branch create a pull request as described above.
2. Any conflicts would be indicated in the pull request webpage as shown below

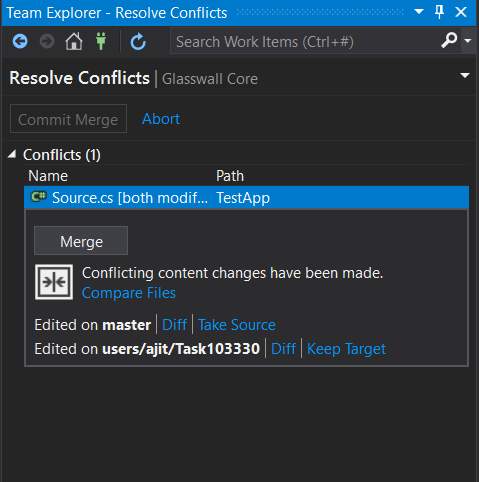


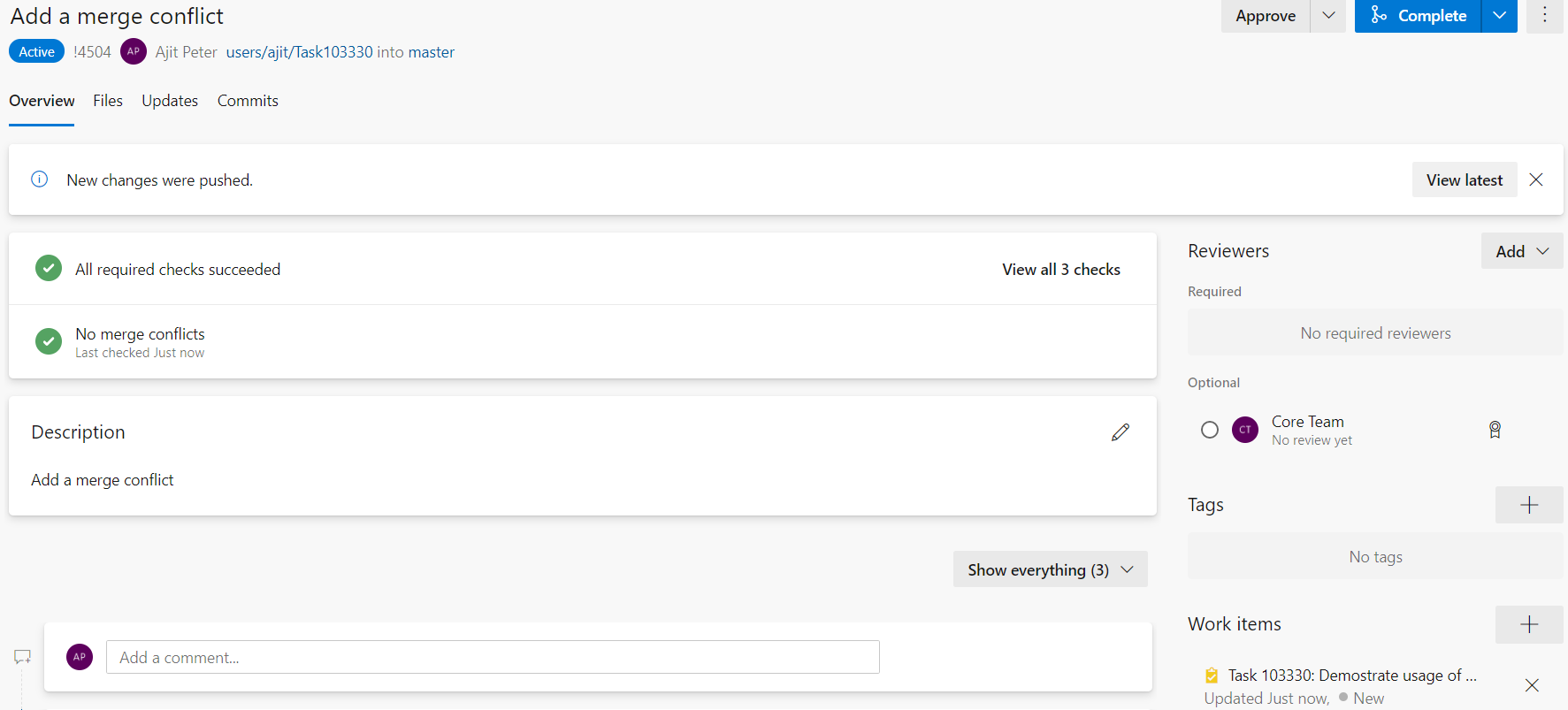
1. The conflict can be resolved by merging “master” branch to your local branch. The option to merge is available from Team Explorer and then select “Branches”



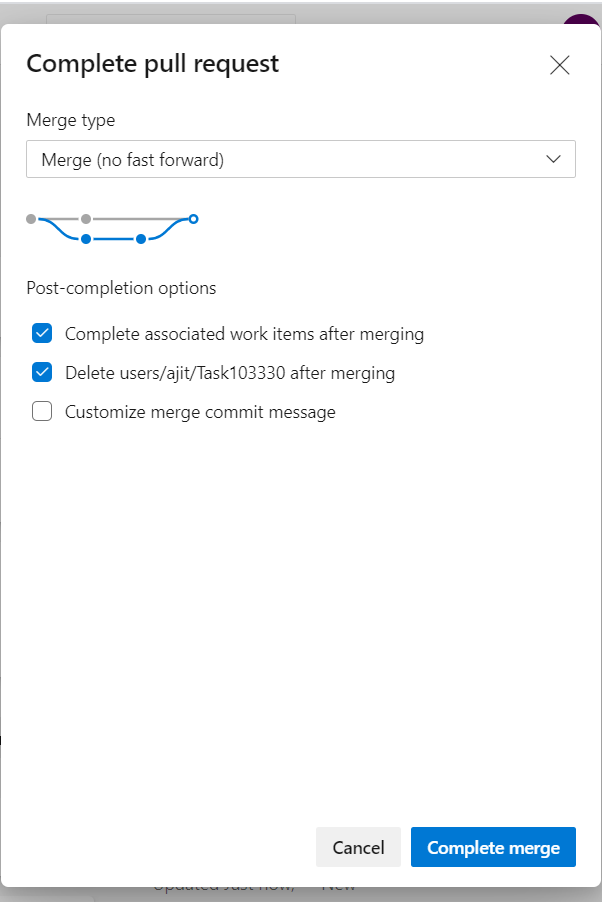
1. Merging the branch would give you the option to resolve the conflicts as shown below.



1. The conflict resolution window presents the source files that have conflicts and clicking on each source file gives the option to merge and resolve the conflict.
2. After resolving the changes commit the changes and push to remote repository. Any new changes will be reflected in the pull request and it would indicate that there are no more conflicts and the code is ready to be merged to “master” branch.



1. Clicking on complete would merge the code into “master” branch and the pull request will be set to complete.



1. Once the merge is complete and the changes are pushed to “master” the following page would be displayed.

