A close up of a logo

Description automatically generatedASSIGNMENT 2: GIT  
OSSW4604

By

Ricardo Gutierrez

D19125375

November 2021

Table of Contents

[GitHub 1](#_Toc88254844)

[Branching and Merging 1](#_Toc88254845)

[Clone and Fork 2](#_Toc88254846)

[Pushing Changes to a Repository to GitHub 3](#_Toc88254847)

[Pull Request 4](#_Toc88254848)

[Dealing With Merge Conflicts 4](#_Toc88254849)

[User Management and Security 5](#_Toc88254850)

# GitHub



Figure 1 Github logo

Github is a platform that is cloud-based where the developers can host their code for software development and version control. It basically allows the developers to manage git repositories online, work and collaborate with other developers. It is the most common platform to host open-source projects because it allows people to work in groups easily.

Although Git and GitHub are two different things, Git is a version control system that uses command line tool and only works locally on your computer whereas GitHub is a hosting service on the web for Git repositories and is a graphical user interface.

# Branching and Merging

Git branching allows developers to work with their code without modifying the main version, what branching does is creating a branch of your code and makes a copy of it, this is helpful so that you can test your code before merging it to the main branch. With regards to merging, merging takes your branch changes and implements them into the main branch.

Text

Description automatically generated

Figure 2 Git Branching Sample

Figure 2 above shows how branching works, first, I created a new branch and switched into this branch with the command “git checkout *name*” after that I edited the README.md and added some new comments then git commit to save the revision.

Text

Description automatically generated

Figure 3 Git Merging Sample

After committing a revision on experiment branch, it’s time to switch to the main branch and then merge the experiment branch into main branch. If you check the README.md you will be able to see the revision locally.

# Clone and Fork

Diagram

Description automatically generated

Figure 4 Clone vs Fork from theserverside.com

Git Clone and Git Fork has the same purpose, creating a copy of a repository but their key difference is that when you clone, it creates a linked copy that will continue to be in sync with the repository – this means that any developer who clones a repository can synchronise their own copy with any revision made by the other developers. Whereas as in Fork, it creates a completely independent copy of the repository which basically means that the developer who performs the fork will have complete control over the newly copied code that means any changes made will not synchronise and those other developers who previously cloned the repository will have no knowledge of the newly forked repository.

# Pushing Changes to a Repository to GitHub

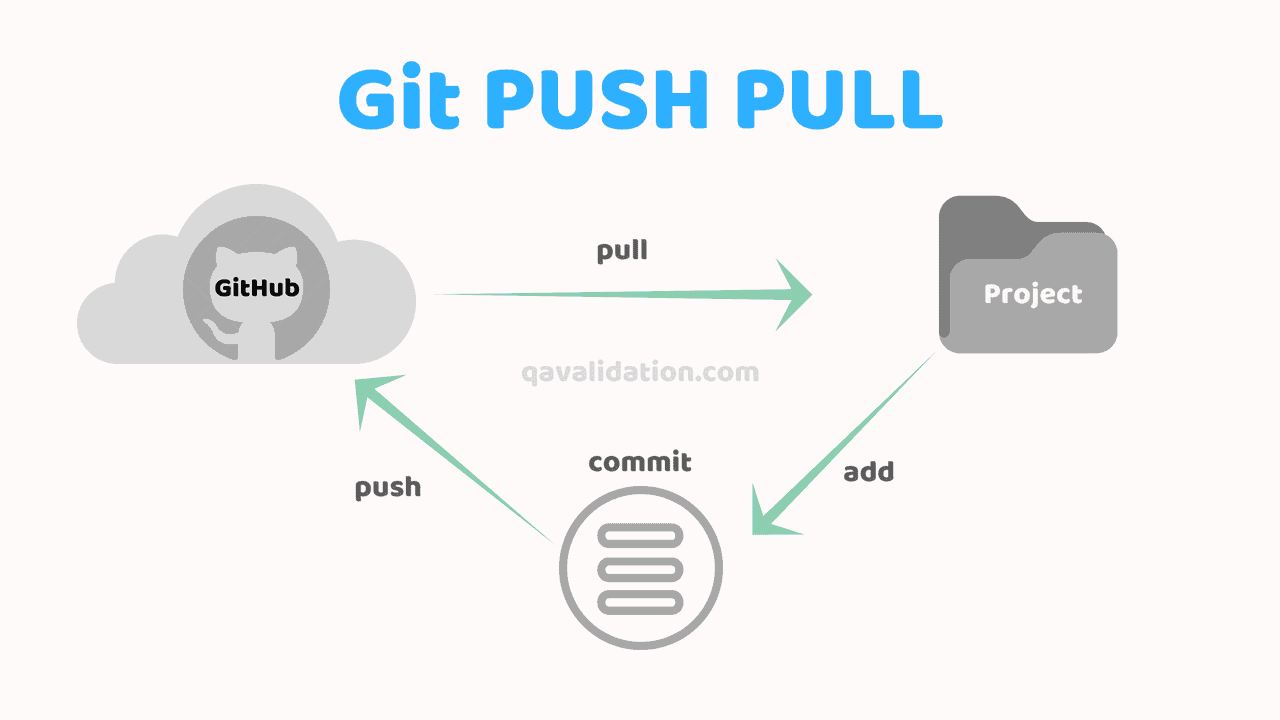


Figure Git Push and Git Pull from qavalidation.com

To push changes to a repository to GitHub can be done by using the *git push* command. Pushing is how you transfer commits locally to a remote repo like GitHub. So basically, after a local repository has been revised, a push is executed to share the revision with other developers or team members.

Text

Description automatically generated

Figure Git Push sample

The Figure 6 above shows the command done to push (upload) the revised code into GitHub. It also shows that username and password was asked, a personal repository token generated by GitHub was used as the password. After that, the revised code will then be pushed and other developers will be able to see the revised code.

# Pull Request

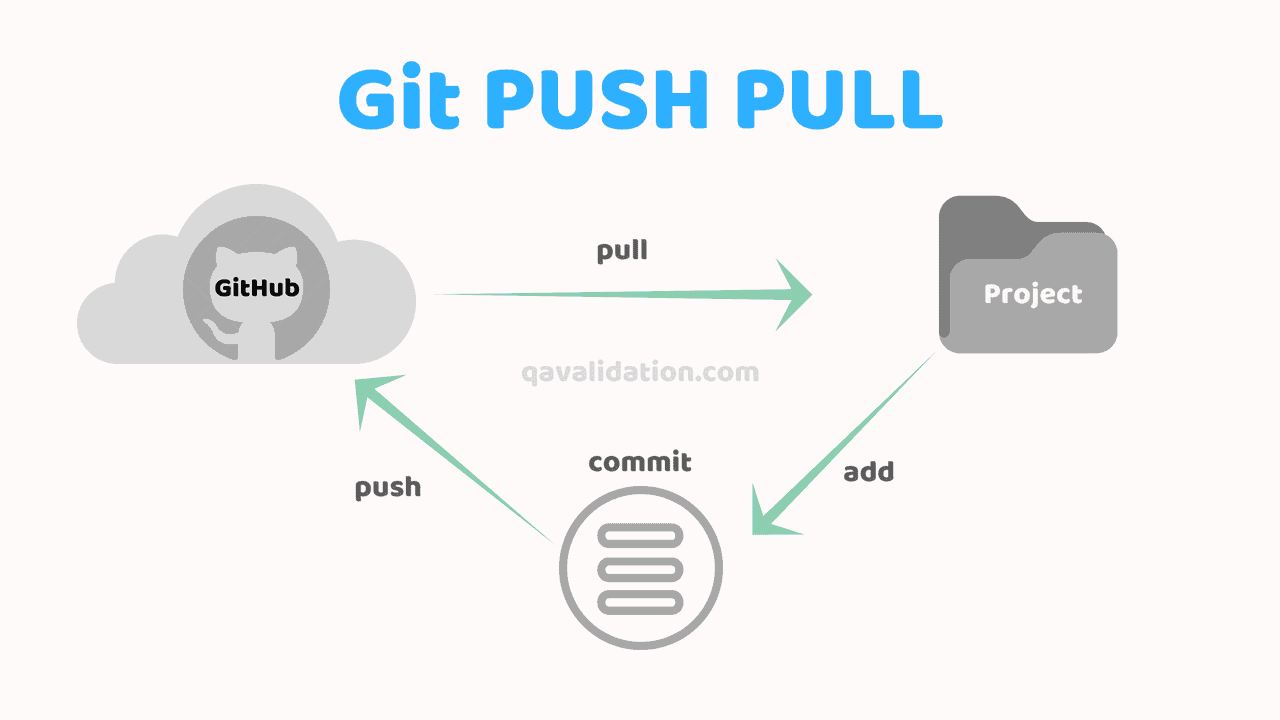


Figure Git Push and Git Pull from qavalidation.com

As we can see from the figure 7 above, pull request basically is you asking a maintainer of a Git repository to discuss and review the potential changes with collaborators and add follow-up commits before your changes are merged into the base branch.

# Dealing With Merge Conflicts

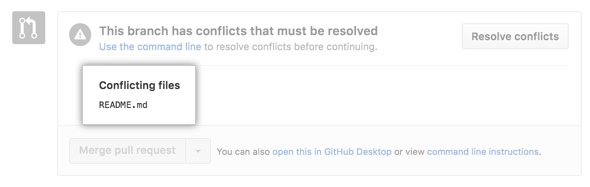


Figure Merge Conflicts Sample from github.com

Based on the documentation from GitHub.com, you encounter merge conflicts when the branches that you have merged has competing commits. An example is when people make different changes to the same line of the same file, or when one person edits a file and another person deletes the same file – Git then will ask for your help to decide which revisions to implement in the final merge.

To deal with a merge conflict, unfortunately, you will have to manually edit the conflicted file to select the revisions that you want to keep in the final merge – which can be done in a couple of ways. First is, If the merge conflict is caused by competing line changes, you can resolve it on GitHub using the conflict editor. Secondly, for all other types of merge conflicts, you will have to deal with this in a local clone of the repository using the command line and push the revision to your branch on GitHub.

# User Management and Security

GitHub has a feature where you can invite other developers to become collaborators to your personal repository. Although you must take note that if you invite other developers, these pending invitations will expire after 7 days. Since you can invite collaborators, you can also remove a collaborator from your personal repository. When you remove a collaborator, they will not be able to read/write access to your repository. Also, if the said developer has created a fork, then that fork will also be deleted.

As with regards to the security, there are some best practices for user security, these includes: 1. Enabling two-factor authentication, 2. Requiring a password manager, and 3. Restricting access to repositories.